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

THE GRADUATE SCHOOL

ADAPTIVE LEAST SQUARES SUPPORT VECTOR REGRESSION FOR HISTORY MATCHING AND MARKOV CHAIN MONTE CARLO UNCERTAINTY QUANTIFICATION

by

Emilio Paulo dos Santos Sousa

A dissertation 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

2019

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

THE GRADUATE SCHOOL

ADAPTIVE LEAST SQUARES SUPPORT VECTOR REGRESSION FOR HISTORY MATCHING AND MARKOV CHAIN MONTE CARLO UNCERTAINTY QUANTIFICATION

by
Emilio Paulo dos Santos Sousa

A DISSERTATION

APPROVED FOR THE DISCIPLINE OF

PETROLEUM ENGINEERING

By Dissertation Committee

Albert C. Reynolds, Chair
Mustafa Onur
Evren M. Ozbayoglu
William Coberly

COPYRIGHT STATEMENT

ABSTRACT

Emilio Paulo dos Santos Sousa (Doctor of Philosophy in Petroleum Engineering)

Adaptive Least Squares Support Vector Regression for History Matching and Markov Chain Monte Carlo Uncertainty Quantification

Directed by Albert C. Reynolds

227 pp., Chapter 5: Conclusions

(471 words)

In the petroleum industry, important decisions need to be made based on uncertain information. To reduce the risks involved, uncertainty quantification of reservoir production forecast is of utmost importance. For practical purposes, application of Bayes' theorem allows the uncertainty in the reservoir model parameters and reservoir predictions to be encapsulated in a posterior probability density function (pdf) for the reservoir parameters. Consequently, the uncertainty quantification problem reduces to the generation of samples from the posterior pdf. The Metropolis-Hastings Markov chain Monte Carlo (MCMC) method provides the means to rigorously sample the posterior pdf. Unfortunately, for our problems MCMC comes with a extremely high associated computational cost, since a large number of states in the Markov chain need to be generated, and each state requires a reservoir simulation run. Many alternative methods have been proposed to approximately sample the posterior, which avoids the high computational cost of MCMC, but none of them is completely rigorous. In this research, we embrace the MCMC method and investigate the feasibility of replacing the reservoir simulator by a fast proxy model, in an attempt to reduce the computational cost of quantifying the uncertainty using MCMC.

The approach adopted here is based on a previous work in which a Gaussian mixture model (GMM) was introduced as an appropriate proposal distribution for MCMC. The GMM

proposal distribution accelerates the convergence of the MCMC method, which in practice translates to a reduction in the required number of reservoir simulation runs. The GMM is constructed centered at modes of the posterior pdf for reservoir parameters.

In this research, we conduct similar procedure, with the important modification that we construct the Markov chain using a proxy model in place of the reservoir simulator to evaluate the Metropolis-Hastings probability of acceptance. For this purpose, we investigate a machine leaning technique, least squares support vector regression (LS-SVR), as a suitable proxy model. The LS-SVR proxy is constructed using a training set. To achieve a reliable proxy model, we introduce a novel procedure to adaptively construct the training set which is used to train the LS-SVR proxy model. Furthermore, we take advantage of the analytical proxy gradient to find modes of the posterior pdf in order to construct the GMM proposal distribution. For large-scale problems, principal component analysis (PCA) is used to reduce the dimension of the problem and the LS-SVR proxy is constructed based on the reduced-order model. Via examples, we show that our proposed method reduced the total number of reservoir simulation runs required for the complete uncertainty quantification process by a factor on the order of 50 to 100. The application examples so far include a single parameter toy problem, an one-dimensional water-flooding reservoir model, a synthetic two-dimensional reservoir model, a three-dimensional model based on the PUNQ-S3 case, and a three-dimensional large scale reservoir model constructed by refining the grid of the PUNQ-S3 model.

ACKNOWLEDGEMENTS

To begin, I would like to thank my advisor, Dr. Albert C. Reynolds, for the inestimable support and invaluable guidance through the last four years. How to approach a problem, how to ask the relevant questions, and how to procedure toward an useful solution, these are the lessons that a take from him for the rest of my life.

I also would like to thank Drs. Mustafa Onur, Evren M. Ozbayoglu and William Coberly for accepting being part of my dissertation committee, besides their insightful comments and support. And how can I forget to thank all faculty members and department staff from The University of Tulsa, specially from the McDougall School of Petroleum Engineering and Department of Mathematics. Many were the courses, many were the lessons, and the most important I will take with me, the knowledge that was transmitted and acquired.

Although I believe we do not have enough time to properly conduct a research and make friends, I was truly happy to find some very special people here in The University of Tulsa. My hearty thanks goes to Christian, Cintia, Elias and Thiana, somehow I think it would not be possible to finish it without your friendship.

I would like to thank Petrobras for all the support, including financial, for the fulfillment of my doctorate.

I would like to thank my family, specially my parents and brothers, for their support.

I dedicate this dissertation to my wife Andrea and my sons Guilherme and Henrique, thanks for your love and patience.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The costs involved in the discovery and development of a hydrocarbon accumulation are extremely high. Several significant decisions on development are made based on partial and limited knowledge. The most important information needed to decide on whether to develop the field and the capital expenditure, needed to do so, is a reasonable accurate estimate of the future production performance of the new discovery. Ultimately, the total amount of possible production and its temporal distribution dictates how the financial resources are applied. To better support decision-making on field development, one must be able to generate a prediction of the future reservoir production and quantify the uncertainty in this prediction. Due to the increasing necessity and commitment to better apply financial resources, increasingly sophisticated tolls for reservoir characterization have been developed. Simplified prediction techniques used in the early days of the oil industry have been replaced by numerical simulators. Currently, the construction of a numerical reservoir simulation model is the standard method to achieve reliability in production forecast for the oil industry.

Unfortunately, virtually all information available to construct such a reservoir model is uncertain. Eventually, dynamic production data become available, and are used to calibrate the original reservoir model. The assimilation of data is called history matching in the petroleum industry. Most commonly, history matching is applied to integrate production data into reservoir models and this problem is the focus of this research. However, information contained in the production data is not enough to resolve all uncertainty. To guide future decisions on the field development and management it is important to quantify the remaining uncertainty in the reservoir model and, most importantly, how the uncertainty is

disseminated to the reservoir production predictions. Accordingly, in the oil industry, uncertainty quantification in reservoir simulation predictions has become a standard procedure to support decisions and mitigate the risks involved in field development

The original motivation for history matching was to procure a sole specific reservoir model for which its pressure and rates predictions were able to match some available dynamic production data with reasonable agreement, and then use this model to make predictions. Current history matching is more nuanced. First, it soon become clear and is now largely recognized that to be considered an acceptable history-matched model, the reservoir model obtained by history matching must conform to the geological model developed for the field from available static data, e.g., seismic data, log and core data, and not simply reproduce the observed data reasonably well. Secondly, it is possible to find several models which are in sufficiently good correspondence with both observed data and some geological model. In fact, it is well established that exist an infinite number of models which give an acceptable history-matched model. As a consequence, the purpose of history matching has shifted to select not just one, but a representative set of models from the total set of plausible reservoir models. The representative ensemble of models should be able to characterize the uncertainty in the reservoir models and future reservoir performance predictions. Within this new history matching scenario, the framework of Bayesian statistics has become an appropriate approach for modern assisted history matching and uncertainty quantification.

In the Bayesian framework for history matching and uncertainty quantification, one needs to provide prior knowledge concerning the uncertain nature of the reservoir model parameters. For practical reasons, the uncertainty in reservoir model parameters is encapsulated into a probability density function (pdf), the so-called prior pdf. Then, Bayes' theorem is used to assimilate the noisy dynamic observed data. For this purpose, the measurement errors contained in the observed data are treated as a random vector with some given pdf. Bayes' theorem results in a pdf for the reservoir model parameters conditioned to the observed data, where this conditioned pdf is the so-called posterior pdf. As discussed earlier, a set of sample from this posterior pdf is used to characterize the uncertainty in the reservoir

model parameters [89]. Consequently, the assisted history matching and uncertainty quantification problem is reduced to the problem of generating samples from the posterior pdf. The predictions using the sampled models are used to quantify the uncertainty in reservoir forecasts, which are then used to support decisions.

As we discuss later in this dissertation, for most practical applications the reservoir simulator represents a non-linear relationship between reservoir model and corresponding predictions. Consequently, the posterior pdf is a non-Gaussian distribution, which can potentially be a multi-modal distribution, and sampling from a non-Gaussian posterior pdf can potentially be computationally expensive. The main goal of this research is to develop a computationally inexpensive framework to sample from the posterior pdf, as we discuss in the remainder of this dissertation.

1.1 Literature Review

Correctly sampling from a multi-modal distribution is a computational expensive task. In the oil industry, great research effort has being employed in the last decades to develop techniques able to produce a meaningful sample from the posterior distribution in a computational feasible manner, as we present in this literature review.

1.1.1 The Randomized Maximum Likelihood Method

In the petroleum industry literature, one of the earliest methods developed to generate samples from the posterior pdf was proposed by Oliver et al. [87]. The method became known as randomized maximum likelihood (RML). A similar sampling method was independently introduced by Kitanidis [64]. To generate samples from the posterior pdf, the RML sampling method starts with unconditional realizations, i.e., samples from the prior pdf, which is assumed to be a Gaussian distribution. Then, the RML performs by conditioning the unconditional realization to a perturbation of the actual observed data. The perturbed observed data is obtained by adding a random realization of the measurement errors to the actual observed data. The measurement errors are assumed to follow a Gaussian distribu-

tion, which is sampled to generate the realizations used to perturb the data. To condition the model, a minimization problem is conducted by minimizing an objective function composed of two quadratic terms, a model mismatch term, which contains the unconditional realization and the prior covariance matrix, and a data mismatch term, which contains the perturbed data and the measurement error covariance matrix. The minimization problem is solved using a gradient-based optimization algorithm, e.g., Gauss-Newton [83, 89] or Levenberg-Marquardt [66, 73, 89]. Oliver [84] and Reynolds et al. [97] presented two distinct proofs that for a linear relationship between reservoir parameters and corresponding predictions, the RML method correctly generate samples from the posterior pdf. Reynolds et al. [97] also further extended the RML method to incorporate uncertainty in the mean of the prior pdf. It is well known that for linear relationship, the posterior pdf is a Gaussian distribution when the prior and measurement errors pdf are assumed Gaussian. Therefore, for the linear case, the RML method essentially transform the unconditional sample from the prior pdf into a conditioned sample from the Gaussian posterior pdf. Unfortunately, there is no guarantee that the resulting sample is indeed a sample from the posterior pdf for the non-linear case which arises in practical applications. Nevertheless, the minimization problem conducted in the RML method always produce samples which come from regions close to the modes of the posterior pdf. For multi-modal distributions, this feature represents a strength of the RML method, i.e., it is expected that the RML method can at least generate a roughly characterization of a multi-modal distribution, see [87, 123, 139].

For practical applications, obtaining the gradient of the objective function to solve the minimization problem required in the RML formulation can be troublesome. To circumvent this problem, Gu and Oliver [49] introduce an ensemble based RML which requires no gradient computation. The method was named ensemble RML (EnRML) and works as an iterative ensemble Kalman filter. Chen and Oliver [21] modified the EnRML to incorporate a Levenberg-Marquardt minimization step (LM-EnRML), which results in better convergence when compared with EnRML. Chen and Oliver [21] also introduced an approximation computationally efficient method in which the model mismatch term in the mini-

mization update step is neglected. The RML method can be understood as an independent Metropolis-Hastings algorithm where the acceptance/rejection step is suppressed [85], which for non-linear cases results in an approximate sampling algorithm. As exposed by Oliver [85], the acceptance/rejection is omitted because is computational prohibitive to compute the so-called Metropolis-Hastings acceptance probability. Oliver [85] introduce an augmented state RML which facilitate the computation of the Metropolis-Hastings acceptance probability. The method proposed by Oliver [85] presented good sampling performance when applied to toy problems with multi-modal posterior distribution. However, the method demands the computation of a Jacobian matrix, which can be computationally intractable for real problems. Stordal and Nævdal [116] introduce a generalized RML (GRML) by generalizing the objective function which is minimized in the original RML. Stordal and Nævdal [116] presented results for a simple weighted objective function in which a weight is applied only to the model mismatch part. The authors presented a theoretical framework to compute the optimal weight for this simple case and showed that the resulting GRML performance is superior to the RML when using the optimal weight. However, computing the optimal weight for real problems can be computationally infeasible.

1.1.2 Ensemble Methods based on the Kalman Filter

In the past few decades, ensemble methods based on the Kalman filter [62] have received intensive attention in both academia and industry. The Kalman filter [62], is a sequential data assimilation algorithm which represents and propagates the uncertainty in the system by means of the first and second order moments, i.e., the mean and covariance matrix, of the associated probability distribution. Consequently, the Kalman filter [62] is more suitable for Gaussian or near Gaussian distribution, which for reservoir petroleum applications translates to the linear case under prior Gaussian distribution. The extended Kalman filter (EKF) is an extension of the Kalman filter [62] for non-linear problem. In EKF, the non-linear model is expanded centered at the current estimate of the state mean, then the expansion is truncated to generate a linear approximate model. However, this

linearization can lead to unbounded error variance growth for highly non-linear problems [31, 32, 42, 13, 77, 34].

For some practical applications, to compute, store and update a full covariance matrix can be so computational expansive, that the application of the Kalman filter [62] and EKF becomes infeasible. To circumvent this limitation, Evensen [33] introduced the ensemble Kalman filter (EnKF), which also performs better for non-linear problems than does EKF. In the EnKF formulation, the uncertainty in the system is represented and propagated by means of an ensemble of system parameters. In this case, when required, the mean and covariance matrix are computed directly from the ensemble, which means that a low rank approximation of the covariance matrix is used. The EnKF method was further developed by Burgers et al. [17], where it was shown that the assimilated observations should be treated as random variables, and Houtekamer and Mitchell [59], which suggested the use of multiples ensembles. The EnKF was first introduced in the petroleum reservoir literature by Nævdal et al. [81] and Nævdal et al. [82]. Aanonsen et al. [2] and Oliver and Chen [86] presented a comprehensive review of EnKF applications in the petroleum literature. The EnKF method assimilates data sequentially in time, which is a suitable feature for petroleum reservoir application where the observed data used to become available at discrete times. To avoid rerun the reservoir simulator from the initial time, each time a new data is assimilated, the EnKF is formulated as a parameter-state estimation problem, i.e., the uncertainty is simultaneously propagated for both the model parameters (such as grid-block porosity and permeability) and model states (such as grid-block pressure and saturation). However, it requires that the updated parameters and model states are statically consistent, i.e., that the statistical properties inferred from the updated ensemble of model states present the same statistical properties computed from the model states obtained running the reservoir simulator from time zero using the updated ensemble of model parameters. However, it can be shown [124] that the statistical consistency only holds for the linear-Gaussian case. Furthermore, the statistical consistency usually does not hold for highly nonlinear petroleum reservoir cases, as shown by Seiler et al. [109] and Wang et al. [136].

To avoid the statistical inconsistency issue, van Leeuwen and Evensen [126] introduced the ensemble smoother (ES). Unlike EnKF, ES is solely a parameter estimation algorithm which does not assimilate data sequentially in time, but rather assimilates all available observed data simultaneously. Although assimilating all data simultaneously prevents the statistical inconsistency between updated parameters and updated states, usually the quality of the data match obtained with ES is inferior to the one with EnKF. Several authors suggested iterative approaches to enhance the quality of the data match [27, 29, 21, 60, 72]. The ensemble smoother with multiple data assimilation (ES-MDA) was introduced by Emerick and Reynolds [27, 28, 29, 30], motivated by the results of Reynolds et al. [99] and Rommelse [105]. The ES-MDA improves the quality of the data matching by assimilating the same data multiple times with the corresponding measurement error covariance matrix multiplied by an inflation factor. Emerick and Reynolds [29] showed that the summation of the inverse of the inflation factors should equal one over all assimilation steps. Both the total number of times that the same data is assimilated and the corresponding inflation factors have to be specified by the user, which can be considered as a drawback of the method. Based on the works of Hanke [52] and Iglesias and Dawson [61], Le et al. [65] suggested to adaptively determine the inflation factors simultaneously with the data assimilation. Recently, Rafiee and Reynolds [94, 93] derived a methodology to compute the inflation factors for ES-MDA which usually delivers as good or a better data match, and at a lower computational cost, than the methods considered in Le et al. [65].

Although, ensemble methods based on the Kalman filter [62] present a computationally efficient framework for petroleum reservoir history matching and uncertainty quantification, for highly nonlinear practical applications, the updated ensemble does not accurately describe the posterior pdf, i.e., a correct uncertainty quantification is not achieved.

1.1.3 Markov Chain Monte Carlo Sampling

A frequently applied and endorsed rigorous technique to sample any given pdf is the Markov chain Monte Carlo (MCMC) method. A Markov chain is a sequence of random

variables, which are conventionally regarded as the *states* of the generated chain. Starting from a random initial state, one can constructs a Markov chain by consecutively appending news states employing some specified proposal mechanism and an acceptance/rejection step. The proposal mechanism is used to propose a candidate for the new (next) state in the chain. Whenever the proposed state is accepted in the acceptance/rejection step, it becomes the next state in the chain and one continues the procedure by proposing a new candidate. When the candidate is rejected, the current state is repeated as the next state in the chain, one again proceeds by proposing a new candidate. Normally, a given probability distribution, the so-called proposal distribution, is used as the proposal mechanism to generate new states in the chain. The distinctive feature of any Markov chain is the enforced property that the probability of proposing a new state in the chain depends only on the current state. Accordingly, the proposal distribution is assumed to depend only on the current state, regardless of the history of previous generated states.

Depending on the chosen proposal mechanism, the resulting Markov chain may present states that repeatedly occur at a fixed period, i.e., the Markov chain inevitably return to a particular state exactly at every nth iterations. Those repeated states are termed as periodic states. A Markov chain which has no periodic states is said to be *aperiodic*. For some proposal mechanism, several probable states are impossible to reach depending on the choice of the initial state. For an aperiodic Markov chain for which it is possible to reach any probable state, with nonzero probability, when starting from any possible state, the given Markov chain is said to be *irreducible*. For a given proposal mechanism, when the probability to move from a particular state to a new state is the same probability to move back from the new state to the original state, the Markov chain is said to be *reversible*. A reversible Markov chain is said to satisfy the detailed balance condition [35, 125]. Whenever a Markov chain is aperiodic, irreducible and reversible, the Markov chain is said to be *ergodic*. It is associated with any given ergodic Markov chain an unique specific probability distribution, the so-called invariant or stationary probability distribution [35, 125]. For a given ergodic Markov chain started at any random state, when the number of generated states in the chain

approach infinity, the probability distribution which is obtained with the generated states approaches the stationary distribution of the corresponding Markov chain, i.e., the generated states represent a sample from the stationary distribution. Naturally, the states generated at the beginning of the chain do not represent the stationary probability distribution and should be discarded. Those initial states form the so-called burn-in period, i.e., the period for which the generated states do not represent samples from the stationary distribution.

The MCMC sampling method performs by generating an ergodic Markov chain which has stationary probability distribution equal to the probability distribution one wishes to sample. In this context, the probability distribution which is being sampled is referred to as the target pdf. After discarding the burn-in period, the remaining states represent a proper set of samples from the target pdf. It has been proved [35, 125] that for an ergodic Markov chain, the associated MCMC sampling method asymptotically samples from the target pdf as the total number of generated states in the chain approaches infinity, provided that the proposal mechanism is able to propose states from any region of nonzero probability.

Numerous developed variants of the MCMC sampling method essentially follows the same procedure. First one proposes a candidate to new state in the Markov chain by sampling the chosen proposal distribution, then accepts or rejects the proposed state based on some well defined acceptance criterion. To make the sampling process feasible for practical applications, normally one assumes some well known and easy to sample distribution as the proposal distribution to generate the candidates to new states in the chain. Historically, the Metropolis-Hastings algorithm [76, 56] is the most applied MCMC sampling method. Metropolis et al. [76] introduced the MCMC method and developed a sampling algorithm with an acceptance/rejection criterion which only depends on ratios of the target pdf. This particular feature of Metropolis et al. [76] algorithm represents a notable advantage for practical applications, since one only needs to know the target pdf up to a normalizing constant. Hastings [56] extended the sampling algorithm proposed by Metropolis et al. [76] by generalizing both the proposal distribution and the acceptance/rejection criterion introduced by Metropolis et al. [76]. Hastings [56] contribution resulted in the framework which is referred

today as the Metropolis-Hasting algorithm.

As discussed earlier, starting from any randomly selected initial state, the Metropolis-Hastings algorithm is guaranteed to converge and asymptotically sample from the target pdf as the number of generated states increase [35, 125]. However, the burn-in period may have a very long duration for some choices of the proposal distribution. For petroleum reservoir applications, the target pdf is represented by the posterior pdf for reservoir model parameters, which for practical applications depends on the predictions of a reservoir simulator. As a consequence, to evaluate the posterior pdf one reservoir simulation run is required. Depending on the proposal distribution, the Metropolis-Hastings algorithm may require hundreds of millions of reservoir simulation runs [44, 71, 28] just to converge to the target pdf, i.e., to start to sample from the target pdf, which here is the posterior pdf for reservoir model parameters. Naturally, for applications where one reservoir simulation run requires more than a couple minutes, sampling the posterior pdf using the Metropolis-Hastings algorithm is computationally prohibitive, although theoretically it is a rigorous method for sampling the posterior pdf.

Motivated by the work of Gao et al. [41], Li and Reynolds [69, 68] introduced a computationally efficient two-level MCMC method based on the Metropolis-Hastings algorithm. Basically, Li and Reynolds [69, 68] proposed the construction of an approximation of the posterior pdf to use as proposal distribution in a Metropolis-Hastings MCMC framework. The underlying idea is since the proposal distribution is close to the target pdf, the Markov chain converges fast, consequently the associated burn-in period is reduced. The approach of Li and Reynolds [69, 68] considerably reduces the total number of required reservoir simulation runs. Li and Reynolds [69, 68] recommended constructing a Gaussian mixture model (GMM) [100, 138], centered at modes of the posterior pdf, to use as proposal distribution. To find modes of the posterior pdf, Li and Reynolds [69, 68] conducted several minimization problems using an in-house reservoir simulator which was able to solve the adjoint problem [20, 19]. This limits the application of their method since commercial reservoir simulators usually are not able to compute the adjoint solution. Rafiee and Reynolds [95, 93] proposed

a modification of Li and Reynolds [69, 68] approach and circumvent the adjoint solution issue by computing an approximate gradient using the distributed Gauss-Newton (DGN) method, which was proposed by Gao et al. [41].

1.1.4 Support Vector Regression

Considerable effort and attention was given to the field of *machine learning* in the last few decades. The term *machine learning*, which was first used by Samuel [107] in his work about the game of checkers, symbolizes the suite of techniques that enable machines to recognize patterns, or more generally learn patterns, from some available data, without being explicit programmed. Nowadays, *machine learning* methods are applied to a variety of practical problems, including the *classification problem*, which denotes the ability to separate a given set of inputs into distinct classes, the *regression problem*, which represents the estimation of future function outcomes based on some already known function outputs, and the *clustering problem*, which means the capacity to recognize different classes from a given data set and then classify the data into the respective classes found.

The support vector machine (SVM), which is a class of learning machines, has become very popular in recent decades due to its reported and recognized good performance and generalization when compared with other available *machine learning* techniques. The SVM method proceeds by employing a suite of inputs and their corresponding output data, conventionally termed as a *training set*, to enable the machine to learn or recognize the unknown patterns which are present in the data. The primordial idea which led to the present form of the SVM algorithm is the work of Vapnik and Lerner [134, 135] on pattern recognition, which is a *machine learning* classification problem. In their work, the authors introduced the *generalized portraits* algorithm, which uses a single-valued transformation to relate the patterns of a given set of images to points on an unit sphere in a given Hilbert space. Vapnik and Chervonenkis [131, 132] further developed the ideas present in the *generalized portraits* algorithm. The pattern predictions represented as a weighted summation of inner products, along with the sparsity representation of the vector of summation weights,

which are prominent features of modern SVM algorithms, were already present in this latter work of Vapnik and Chervonenkis.

The field called *Statistical Learning Theory* (also known as *Vapnik–Chervonenkis theory* or *VC theory*) arises from the work of Vapnik and Chervonenkis [133]. In a further development of the *Statistical Learning Theory*, Vapnik [127, 128] presents the original SVM formulation for the case of *linearly separable data*. In *machine leaning*, the *linearly separable data* designate the case in which a collection of points, or vectors, belonging to two distinct classes, is endowed with *linear separability* in a given space, i.e., the collection of points can be completely and unequivocally separated into its two distinct categories by means of a hyperplane. In the linear SVM classifier presented by Vapnik [127, 128], one employs a given *training set*, i.e., a suite of input vectors and their corresponding output classifications, to construct a hyperplane to divide the given data into distinct categories. The distinctive idea introduced by Vapnik was to uniquely define an optimal hyperplane by maximizing the distance between the optimal hyperplane and the input vectors belonging to a given training set. For any vector in a given vector space, a hyperplane can be defined by the inner product between the given vector and a vector of weights, plus a bias term. To classify any vector using a particular vector of weights and bias term, one proceeds by computing the inner product between the vector and the vector of weights, and then summing the result to the bias term. If this computation results in a positive number, the vector being classified is on one side of the hyperplane, and thus belongs to a determined class. Conversely, if this computation results in a negative number, the vector being classified is on the opposite side of the hyperplane, therefore belongs to the other class. Vapnik [127, 128] showed that maximizing the distance between the optimal hyperplane and the vectors belonging to a given training set is equivalent to minimizing the Euclidean norm of the corresponding hyper-plane's vector of weights, subject to the constraint that the product between the predicted classification using the resulting optimal hyperplane for all the vectors in the training set and their respective true classification is greater or equal to unity. The proposed formulation results in a quadratic programming problem that one solves to find the vector

of weights which defines the optimal hyperplane. The minimization problem is usually first formulated in the primal space of the weights, then converted to the dual space of Lagrange multipliers. For any given training set, the resulting dual problem has a global unique solution, which is a desirable feature for any minimization problem, and represents an important advantage of the SVM formulation over other *machine learning* techniques. The resulting vector of weights, which defines the optimal hyperplane, is given as a linear combination of the vectors in the training set, using the respective Lagrange multipliers as weights in the linear combination. Therefore, the optimal hyperplane is given as a weighted summation of inner products between the vectors in the training set and the vector one wants to classify, with the weights given by the respective Lagrange multipliers. In the proposed formulation, several of the computed Lagrange multipliers vanish. Furthermore, the nonzero Lagrange multipliers correspond to the vectors in the training set which are closer to the resulting optimal hyperplane. Thus, only a few vectors, among those originally present in a given training set, are required to uniquely define the optimal hyperplane. This sparsity representation constitutes another important advantage of the SVM formulation proposed by Vapnik. The small number of vectors with nonzero Lagrange multipliers are the so-called *support vectors*, and are the only vectors used to build the final representation of the optimal hyperplane. Anderson and Bahadur [7] further developed the SVM algorithm and presented a SVM formulation closer to its current form.

Cortes and Vapnik [24] extended the SVM algorithm for the *linearly non-separable data* case, i.e., the case where there is some overlap between the points of two distinct categories for a given training set, and although one could still construct a hyperplane that approximately separate the data, points close to the hyperplane can be misclassified. For the *linearly non-separable data* case, the SVM formulation of Vapnik [127, 128] is not feasible because no hyperplane is capable of completely and unequivocally separating the data. Cortes and Vapnik [24] circumvent this limitation by adding additional slack variables to the SVM formulation and enabling misclassifications in the final predictions. This approach is known as the *soft margin classifier*.

Perhaps one of the most important and recognized feature of the SVM method is it relatively easy extension to the nonlinear case, which ensures broad practical applicability of the method. Vapnik [129, 130] extend the SVM algorithm for the nonlinear case, i.e., for the cases where a hyperplane is not capable of separating the data and one needs to resort to some nonlinear separation hyper-surface. For the nonlinear case, Vapnik [129, 130] proposed to first transform the vectors in the training set to some high-dimensional space, the so-called *feature space*, by means of a nonlinear mapping, and then simply apply a linear SVM in the feature space. The optimal hyperplane constructed in the *feature space* projects into a hyper-surface on the original input space which is capable of separating the given data. The underlying key idea is to choose a nonlinear mapping to a *feature space* where the given data presents linear separability [92, 25, 115]. Some theoretical results [92, 25, 115] support the existence of such a nonlinear mapping. However, a clear procedure of how to derive this nonlinear mapping is yet to be discovered.

As discussed above, the nonlinear SVM formulation resembles the formulation for the linear SVM, the main difference is that, with the nonlinear SVM, the input vectors in the training set are first mapped to the *feature space*. As a consequence, the optimal hyperplane constructed in the *feature space* requires only inner products of vectors in the *feature space*. Aizerman et al. [4, 5] established a geometric interpretation of positive-definite kernel functions as inner products in some high-dimensional feature space. A positive-definite kernel function is any symmetric continuous function that satisfies Mercer's condition [75]. In the formulation proposed by Vapnik [129, 130], there is no need to explicitly compute the nonlinear mapping from the input space to the *feature space*, one can simply employ a kernel function to compute the required inner products in the *feature space*. This replacement of the inner products in the *feature space* by the computation of some kernel function became known as the *kernel trick*, and is one of the main advantages of the nonlinear SVM machines. In its current formulation, the SVM algorithm represents a nonlinear generalization of the *generalized portraits* algorithm developed earlier by Vapnik and Lerner [134, 135]. A detailed tutorial on SVM was presented by Burges [18].

Vapnik [129, 130] also extended the SVM algorithm to the *regression problem*, a method that has become known as the support vector regression (SVR). In the SVR algorithm, Vapnik [129, 130] introduced the so-called Vapnik's ϵ-insensitive loss function. The SVR algorithm proceeds by building a virtual region with radius ϵ around the given data, where ϵ is the parameter from the ϵ-insensitive loss function. The resulting regression function represents the smoothest function that can be enclosed by the virtual region of radius ϵ. For the cases where a virtual region cannot envelop the given data, Vapnik [129, 130] proposed the use of additional slack variables, which is a similar approach of the *soft margin classifier*. Vapnik [129, 130] SVR formulation also results in a sparsity representation of the unknown function delineated by the data, and the input vectors that are used to build the regression function are also called support vectors. A comprehensive tutorial in SVR was presented by Smola and Schölkopf [112].

Suykens and Vandewalle [118] and Suykens et al. [121] proposed a modification of the Vapnik [129, 130] SVM formulation by employing a squared loss function in the minimization problem formulation and replacing the inequality constraint in the Vapnik [129, 130] formulation by an equality constraint. The authors refer to this approach as the least squares support vector machine (LS-SVM). By using an equality constraint in the minimization problem, in the LS-SVM classifier the quadratic programming problem, which is solved in the original Vapnik [129, 130] formulation, reduces to solving a linear system of equations with dimension equal to the size of the given training set. To solve a linear system of equation can be easier for several practical applications. However, the sparsity representation is lost in the LS-SVM classifier. Hence, all vectors in the training set become support vectors, i.e., all vectors in a given training set are used to construct the optimal hyperplane. Suykens and Vandewalle [119] and Suykens [117] extended the LS-SVM to the regression case, method called the least squares support vector regression (LS-SVR). As pointed by Suykens [117], the LS-SVR formulation represents a *ridge regression* [57] applied in the *feature space*.

1.2 Research Objectives

As stated in the literature review, uncertainty quantification in reservoir production forecasts using MCMC may requires an impractically large number of reservoir simulation runs to sample the target pdf. Consequently, applying MCMC for real reservoir petroleum applications may be computationally prohibitive, even with the innovations of Li and Reynolds [69] and Rafiee and Reynolds [95]. The overriding objective of this research is to develop a methodology that requires ten thousand or less reservoir simulation runs to characterization of the posterior pdf with the Metropolis-Hastings MCMC sampling algorithm.

To obtain sufficient computational efficiency to promote the use of MCMC for uncertainty quantification, this research investigates the use of a proxy model as a replacement for the reservoir simulator. The proxy model investigated is the least-squares-support-vector-regression (LS-SVR) model. The proxy model is given as an analytical expression which allows one to directly compute the reservoir predictions, necessary to evaluate the posterior pdf for a given reservoir model, up to the normalizing constant, without running the reservoir simulator. As knowledge of the posterior pdf's normalizing constant is not required to evaluate the acceptance probability of the Metropolis-Hastings MCMC sampling algorithm, MCMC can be applied to characterize the posterior pdf without ever running the reservoir simulator. As an evaluation of the analytical expression for the LS-SVR proxy requires a tiny fraction of the time required to run the full reservoir simulation model, the cost of generating Markov chains is relatively small, even if it proves necessary to construct chains of lengths on the order of tens of thousands to several millions for the chain to converge to the posterior pdf.

The first step of the proposed approach resembles one used in the works of Li and Reynolds [69] and Rafiee and Reynolds [95], i.e., we first construct an approximation of the posterior, then we use it as the proposal distribution in a Metropolis-Hastings MCMC framework. We use a Gaussian mixture model (GMM) [100, 138] to approximate the posterior pdf. We construct the GMM approximation centered at modes of the posterior pdf. However, the

procedure we use to find modes is completely different than those of Li and Reynolds [69] and Rafiee and Reynolds [95]. To find the modes, we solve several minimization problems, starting at different initial guesses. To avoid the need of the adjoint solution, we propose instead to use the gradient information from the LS-SVR proxy model. To achieve good accuracy in the proxy predictions, we propose to solve all minimization problems simultaneously and use the model updates to iteratively enhance the proxy prediction. After finding the modes needed to construct the GMM approximation, we obtain a computationally efficient and sufficiently accurate LS-SVR proxy model which is used to replace the reservoir simulator when generating the Markov chains, consequently, our algorithm does not require any reservoir simulation runs to construct the Markov chain in order to characterize the posterior pdf of reservoir model parameters conditioned to production data. Thus, uncertainty quantification requires no runs of the reservoir simulator, only evaluations of the analytical form of the LS-SVR which is very fast. Training of the LS-SVR proxy may require a few thousand reservoir simulation runs, but this is computationally feasible. The main contribution of this research is the development of methodology based on the LS-SVR proxy to makes the application of MCMC practical for realistic problems.

1.3 Dissertation Organization

This dissertation is organized into five chapters and two appendices. In this Chapter 1, we present a statement of the history matching and uncertainty quantification problem, a pertinent literature review of the randomized maximum likelihood method, ensemble methods originated from the Kalman filter, Markov chain Monte Carlo sampling and support vector regression, and state the research objectives. In Chapter 2, we discuss the mathematical background for the Bayesian framework for history matching and uncertainty quantification, the Gaussian mixture model as an approximation of the posterior pdf, and the Metropolis-Hastings Markov chain Monte Carlo sampling algorithm. In Chapter 3, we present the least-squares-support-vector-regression formulation and discuss the influence of its training parameters, we present our proposed methodology which consists of two steps,

firstly, we conduct several minimization problems to find modes of the posterior pdf, then we use the modes to construct a Gaussian mixture model approximation of the posterior pdf, secondly, we use the Gaussian mixture model approximation as the proposal distribution to construct a Markov chain using the proxy model, we also present how to apply the proposed methodology to large scale problems and present a tentative of improve the proxy predictions using analytical solutions. In Chapter 4, we present applications of the developed method for five cases, a toy problem, an one-dimensional reservoir model, a two-dimensional reservoir model, the PUNQ-S3 case, and a large scale model based on the PUNQ-S3 case. In Chapter 5, we present the conclusions. In Appendix A, we present the training procedure for the least squares support vector regression proxy model. Finally, in Appendix B, we discuss the trust region minimization algorithm which is used to find modes of the posterior pdf.

CHAPTER 2

MATHEMATICAL BACKGROUND

As discussed in the Chapter 1, the oil industry relies on reservoir simulation predictions to assess the reservoir future performance. The predicted performance support the key decisions related to the development of an oil field. Unfortunately, the simulation models, which are used to evaluate the reservoir future performance, are uncertain. One must consider the influence of this uncertainty in the reservoir simulation predictions. In this Chapter 2, we introduce and discuss some of the practical approaches adopted by the industry to quantify this uncertainty, with focus on those embedded in the theoretical framework of Bayesian statistics.

2.1 History Matching and Uncertainty Quantification in a Bayesian Framework

Normally, an initial reservoir simulation model is constructed using available static data. The static data usually include a geological model developed for the oil field, seismic-based reservoir geometry, reservoir rock petrophysical properties derived from seismic inversion data and calibrated with log and core data from drilled wells, and reservoir fluid properties acquired from reservoir fluid samples collected during well testing procedures. Most information employed to construct the initial reservoir model is uncertain. Some uncertainty remains throughout the development cycle of the field. To rely on the reservoir simulator predictions as an instrument to support decisions, one must quantify how the uncertainty in the initial information is propagated to the reservoir simulator predictions.

As dynamic data from drilled wells became available, the initial reservoir model is modified and calibrated so that its predictions became consistent with the observed dynamic

data. The assimilation of dynamic data into the reservoir simulation model is called history-matching. In the oil industry, history matching is the main technique employed to reduce the uncertainty in the reservoir simulator predictions. Hence, in a reservoir history-matching framework, given a mathematical representation of the fluid flow in the porous media, which we regard here as the *forward model*, and some measurable reservoir response, regarded as the *observed data*, one seeks to determine which physical properties of the reservoir, regarded as a *model*, enables the reservoir simulator predictions to reproduce the measured reservoir response.

Originally, the basic idea of history matching was to find a particular reservoir model so that the forward model predictions were in good agreement with a given observed data. Modern history matching is more nuanced. First, it is now well established that for a reservoir model to be considered an acceptable history-matched model, it must not only give predicted data in reasonable agreement with observed data, but the model must also be consistent with seismic data, log data, core data or more generally a geological model developed from static data. Secondly, it is now widely recognized that there exist numerous models, in fact an infinite number, that are in acceptable agreement with both observed data and some geological model. In practice, due to limited and noisy observed data, one is unable to determine the real nature of the reservoir, therefore the true reservoir model is never completely known. Thereby, the main objective has now switched to select a representative ensemble of models from the total set of plausible reservoir history-matched models. The selected ensemble can be used to provide a meaningful characterization of the uncertainty in the reservoir properties, and perhaps more importantly, the reservoir simulator predictions from the selected ensemble of models can provide an useful quantification of uncertainty in future reservoir performance prediction. For the uncertainty characterization problems addressed in this dissertation, the approach of Bayesian statistics is appropriate [122, 89].

The history-matching problem of finding a reservoir model that allows the reservoir simulator to reproduce within the noise level the observed reservoir response is an inverse problem. Inverse problems are usually ill-posed and in particular, for problems addressed in

this dissertation, generally have multiple solutions. Here, the history matching problem is approached from the Bayesian point of view. Consequently, instead of looking for a particular best solution for the reservoir model which reproduces the observed reservoir response, we characterize the solution of the inverse problem by means of a probability density function (pdf) for the reservoir model parameters, conditioned to the observed data. The conditional pdf is the so-called posterior pdf for the reservoir model parameters, whereas pdf for the reservoir model parameters, which is constructed from static data, is referred to as the prior pdf and is assumed to be known. In this context, the quantification of uncertainty in the reservoir future performance reduces to the characterization of the posterior pdf for the reservoir model parameters. In practice, the characterization of the posterior pdf translates to the generation of a sample from this pdf, which results in a suite of realizations containing highly probable and history-matched reservoir models that honor the observed data and are also consistent with the prior reservoir model. Reservoir simulation runs are performed using the generated samples, then the resulting predictions are used to quantify the uncertainty in future reservoir performance.

Throughout, we represent the properties of the reservoir model, which will be calibrated to dynamic data, by the N_m-dimensional column vector of model parameters $\boldsymbol{m}$, and the forward model for the history-matching period by some function "g_h" acting on the model $\boldsymbol{m}$, the predicted reservoir simulator response corresponding to the observed data, for a given model $\boldsymbol{m}$, is given by

$$\boldsymbol{d}_h = g_h(\boldsymbol{m}) \, . \tag{2.1}$$

In Eq. 2.1, the $N_{d_h}-$dimensional column vector $\boldsymbol{d}_h$ represents the predicted data for a given particular model $\boldsymbol{m}$, which corresponds to observed data collected during the history-matching period.

To simplify the formulation and enable a formal mathematical treatment of the problem, we make the normal assumption that the prior pdf for the uncertain vector $\boldsymbol{m}$ can be well approximated by a multivariate Gaussian distribution, with prior mean denoted by $\boldsymbol{m}_{\rm pr}$

and prior $N_m \times N_m$ covariance matrix denoted by C_M, i.e., $\boldsymbol{m} \sim \mathcal{N}(\boldsymbol{m}_{\text{pr}}, C_M)$, with pdf $f(\boldsymbol{m})$ given by

$$f(\boldsymbol{m}) = \frac{1}{\sqrt{(2\pi)^{N_m}\,|C_M|}}\,\exp\left[-\frac{1}{2}(\boldsymbol{m}-\boldsymbol{m}_{\text{pr}})^T\,C_M^{-1}\,(\boldsymbol{m}-\boldsymbol{m}_{\text{pr}})\right]. \tag{2.2}$$

In Eq. 2.2 and throughout this dissertation, $|\cdot|$ denotes the determinant of a given matrix. Thus, in Eq. 2.2, $|C_M|$ denotes the determinate of C_M. Throughout this dissertation, the superscript "T" represents the transpose of a given matrix or vector, i.e., $(\boldsymbol{m}-\boldsymbol{m}_{\text{pr}})^T$ in Eq. 2.2 represents the transpose of the difference between the vectors $\boldsymbol{m}$ and $\boldsymbol{m}_{\text{pr}}$. Also in Eq. 2.2 and throughout, the superscript "-1" represents the inverse of a given matrix, that is, C_M^{-1} represents the inverse of the matrix C_M.

Normally, both $\boldsymbol{m}_{\text{pr}}$ and C_M are assumed known. The prior mean $\boldsymbol{m}_{\text{pr}}$ usually represents an expectation for the reservoir properties based on static data, or, when little data is available, based on knowledge from similar oil fields. The prior covariance C_M is usually constructed based on a previously selected covariance function which is also estimated from static data.

Assuming that the function $g_h(\boldsymbol{m})$ truly describes the response of the real reservoir during the history-matching period, and that the properties of the reservoir can be properly parameterized by the model vector $\boldsymbol{m}$, i.e., assuming that the modeling errors are negligible, the $N_{d_h}-$dimensional column vector of actual observed data, $\boldsymbol{d}_{h,\text{obs}}$, is given by

$$\boldsymbol{d}_{h,\text{obs}} = g_h(\boldsymbol{m}_{\text{true}}) + \boldsymbol{\xi}_{\text{true}}\,. \tag{2.3}$$

In Eq. 2.3, $\boldsymbol{m}_{\text{true}}$ represents the vector of true model parameters and $\boldsymbol{\xi}_{\text{true}}$ represents the true $N_{d_h}-$ dimensional column vector of measurement errors. In this dissertation, following Oliver et al. [89], it is further assumed that the vector of measurement errors follows a multivariate Gaussian distribution, with zero mean and $N_d \times N_d$ covariance matrix C_D, i.e., $\boldsymbol{\xi} \sim \mathcal{N}(\boldsymbol{0}, C_D)$. For production data measured at monthly intervals, the measurement errors are usually assumed to be independent, and thus the covariance matrix C_D is a diagonal

matrix. Several authors [1, 111, 141, 30] have propose ways to estimate C_D. Emerick and Reynolds [30] actually applied a procedure to estimate covariance functions to estimate the covariance function of measurements errors necessary to construct C_D for both production data, where C_D turned out to be diagonal, and for a seismic data, where C_D is generally not diagonal. For water rate data, Emerick and Reynolds [30] found that measurement errors tend to be proportional to the magnitude of the actual measurement acquired and we generally use this assumption in the computational examples presented later in this dissertation.

In the Bayesian approach for the history-matching inverse problem, one assumes that some prior knowledge about the uncertain nature of the model $\boldsymbol{m}$ is known. Here, this prior information is represented by the prior pdf given by $f(\boldsymbol{m})$ in Eq. 2.2. Furthermore, in the Bayesian approach the posterior pdf for the model $\boldsymbol{m}$, i.e., the pdf conditioned to the actual observed data $\boldsymbol{d}_{h,\text{obs}}$, denoted by $f(\boldsymbol{m} \mid \boldsymbol{d}_{h,\text{obs}})$, or simply by $\pi(\boldsymbol{m})$, is given by Bayes' theorem as

$$\pi(\boldsymbol{m}) \equiv f(\boldsymbol{m} \mid \boldsymbol{d}_{h,\text{obs}}) = \frac{f(\boldsymbol{d}_{h,\text{obs}} \mid \boldsymbol{m})\, f(\boldsymbol{m})}{f(\boldsymbol{d}_{h,\text{obs}})} = a_d\, \mathcal{L}(\boldsymbol{m} \mid \boldsymbol{d}_{h,\text{obs}})\, f(\boldsymbol{m})\,. \tag{2.4}$$

In Eq. 2.4, $\mathcal{L}(\boldsymbol{m} \mid \boldsymbol{d}_{h,\text{obs}}) \equiv f(\boldsymbol{d}_{h,\text{obs}} \mid \boldsymbol{m})$ represents the likelihood function of the model $\boldsymbol{m}$ given a specific observed data vector $\boldsymbol{d}_{h,\text{obs}}$; again $f(\boldsymbol{m})$ denotes the prior pdf for the model $\boldsymbol{m}$, and a_d is the normalizing constant.

We have assumed that the measurement error vector $\boldsymbol{\xi}$ in Eq. 2.3 is multivariate Gaussian with zero mean and covariance matrix C_D, therefore, from Eq. 2.3 we get

$$\mathcal{L}(\boldsymbol{m} \mid \boldsymbol{d}_{h,\text{obs}}) = \frac{1}{\sqrt{(2\pi)^{N_d}\,|C_D|}}\, \exp\left[-\frac{1}{2}\Big(g_h(\boldsymbol{m}) - \boldsymbol{d}_{h,\text{obs}}\Big)^T\, C_D^{-1}\, \Big(g_h(\boldsymbol{m}) - \boldsymbol{d}_{h,\text{obs}}\Big)\right]\,. \tag{2.5}$$

Using Eqs. 2.2 and 2.5, the posterior pdf $\pi(\boldsymbol{m})$ in Eq. 2.4 becomes

$$\pi(\boldsymbol{m}) \equiv f(\boldsymbol{m} \mid \boldsymbol{d}_{h,\text{obs}}) = a_d^*\, \exp\big[-O(\boldsymbol{m})\big]\,, \tag{2.6}$$

where a_d^* is the normalizing constant, and the so-called objective function $O(\boldsymbol{m})$ is given by

$$
\begin{aligned}
O(\boldsymbol{m}) = &+ \frac{1}{2}(\boldsymbol{m} - \boldsymbol{m}_{\text{pr}})^T \, C_M^{-1} \, (\boldsymbol{m} - \boldsymbol{m}_{\text{pr}}) \\
&+ \frac{1}{2}\Big(g_h(\boldsymbol{m}) - \boldsymbol{d}_{h,\text{obs}}\Big)^T \, C_D^{-1} \, \Big(g_h(\boldsymbol{m}) - \boldsymbol{d}_{h,\text{obs}}\Big) \, .
\end{aligned} \tag{2.7}
$$

From the Bayesian point of view, a sampling from the posterior pdf $\pi(\boldsymbol{m})$ of Eq. 2.6 represents, in some sense, the solution of the history-matching inverse problem [89]. A set of samples from the posterior pdf characterizes the uncertainty in the model parameters $\boldsymbol{m}$, conditioned to observed data, and is used to quantify the uncertainty in future forecasts computed using the forward model $g_h(\boldsymbol{m})$, which is represented by a numerical reservoir simulator in the petroleum industry.

The objective function $O(\boldsymbol{m})$ of Eq. 2.7 is the summation of two terms, a model mismatch term which originates from the prior pdf, and a data mismatch term which originates from the pdf for measurement errors. As discussed earlier, a history-match model must provide reservoir predictions in a good agreement with the observed data. Hence, the data mismatch part of the objective function evaluated for a history-match model results in a small numerical value. How small this value should be to consider a model as a history-match model is often not rigorously defined in practice. Besides, the history-matched model must still represent a plausible sample from the posterior pdf, which means that the history-matched model should provide a small value of the model mismatch part of the objective function. Thus, in practice, finding a history-matched model means finding a minimum of $O(\boldsymbol{m})$ of Eq. 2.7. In this sense, the model mismatch term behaves as a regularization term in the minimization problem designed to find a model which allows the reservoir simulator to reproduce the observed data, which mitigates the ill-posedness of the history-matching inverse problem [93].

Most minimization approaches applied to find minima of the objective function $O(\boldsymbol{m})$ of Eq. 2.7 require the gradient and the Hessian of the objective function. From Eq. 2.7, the gradient, $\nabla_{\boldsymbol{m}} O(\boldsymbol{m})$, and the Hessian, $\mathcal{H}(\boldsymbol{m})$, of the objective function $O(\boldsymbol{m})$ are given,

respectively, by

$$\nabla_{\boldsymbol{m}} O(\boldsymbol{m}) = C_M^{-1}(\boldsymbol{m} - \boldsymbol{m}_{\rm pr}) + G_h(\boldsymbol{m})^T C_D^{-1} \Big(g_h(\boldsymbol{m}) - \boldsymbol{d}_{h,\rm obs} \Big) \tag{2.8}$$

and

$$\mathcal{H}(\boldsymbol{m}) = C_M^{-1} + G_h(\boldsymbol{m})^T C_D^{-1} G_h(\boldsymbol{m}) \, . \tag{2.9}$$

In Eqs. 2.8 and 2.9, the $N_{d_h} \times N_m$ matrix $G_h(\boldsymbol{m})$ denotes the so-called sensitivity matrix, which is given by

$$G_h(\boldsymbol{m}) = \Big[\nabla_{\boldsymbol{m}} \Big(g_h(\boldsymbol{m})^T \Big) \Big]^T \, . \tag{2.10}$$

Consequently, the entry in the ith row and jth column of the sensitivity matrix $G_h(\boldsymbol{m})$ represents the partial derivative of the ith predicted datum (i.e., the ith entry of the data vector $\boldsymbol{d}_h$) with respect to the jth model parameter (i.e., the jth entry of the model vector $\boldsymbol{m}$). For large scale reservoir history-matching problems, a computationally efficient and sufficient accurate method to compute the sensitivity matrix $G_h(\boldsymbol{m})$ for a particular reservoir model $\boldsymbol{m}$ is to solve the adjoint problem [20, 19, 67, 98, 89]. In the adjoint formulation, basically one needs to solve two systems of equations. First one solves the flow equations forward in time to determine the reservoir predictions. During the solution of the flow equations, the Jacobian matrix (or the values of the primary simulator variables necessary to compute the Jacobian), at each time step, is preserved. Then, one solves the adjoint equations backward in time to compute the sensitivity matrix. The main coefficient matrix of the adjoint equations is given by the transpose of the Jacobian matrix at each time step [67, 98]. Unfortunately, the majority of the commercial reservoir simulators adopted by the oil industry are not capable of solving the adjoint problem. For practical applications, one must resort to approximate techniques to determine the sensitivity matrix, as we discuss later in this dissertation.

For petroleum reservoir applications, the vector of reservoir model parameters, $\boldsymbol{m}$, incorporates different reservoir properties [98]. Normally, these properties have different

numerical scales, thus, their variances also have different numerical scales. Consequently, the Hessian of the objective function can be ill conditioned, which impacts the convergence of the adopted algorithm used for minimizing the objective function. To alleviate this issue, the following normalization of the vector $\boldsymbol{m}$ can be applied

$$\hat{\boldsymbol{m}} = C_M^{-1/2} \left(\boldsymbol{m} - \boldsymbol{m}_{\text{pr}} \right) . \tag{2.11}$$

In Eq. 2.11, $\hat{\boldsymbol{m}}$ represents the dimensionless counterpart of the vector $\boldsymbol{m}$, and $C_M^{-1/2}$ denotes the inverse of the square root of the prior covariance matrix C_M. The calculation of $C_M^{-1/2}$ is computationally expensive for large scale problems. Usually, it is assumed that a computationally feasible calculation of $C_M^{-1/2}$ can be accomplished by Cholesky decomposition [6, 89]. The adopted minimization algorithm is then conducted using the dimensionless vector $\hat{\boldsymbol{m}}$, instead of the original vector $\boldsymbol{m}$.

For similar reasons, the vector of predicted data corresponding to the history-matching period, $\boldsymbol{d}_h$, is also normalized as

$$\hat{\boldsymbol{d}}_h(\hat{\boldsymbol{m}}) = C_D^{-1/2} \left(g_h(\boldsymbol{m}) - \boldsymbol{d}_{h,\text{obs}} \right) . \tag{2.12}$$

In Eq. 2.12, $\hat{\boldsymbol{d}}_h(\hat{\boldsymbol{m}})$ represents the dimensionless counterpart of $\boldsymbol{d}_h = g_h(\boldsymbol{m})$, and $C_D^{-1/2}$ denotes the inverse of the square root of the measurement error covariance matrix C_D. As discussed earlier, C_D is usually assumed to be diagonal for production data, therefore, the computation of $C_D^{-1/2}$ is straightforward.

Using the dimensionless transformations of Eqs. 2.11 and 2.12, the objective function $O(\boldsymbol{m})$, given by Eq. 2.7, can be rewritten as

$$O_D(\hat{\boldsymbol{m}}) = \frac{1}{2} \hat{\boldsymbol{m}}^T \hat{\boldsymbol{m}} + \frac{1}{2} \hat{\boldsymbol{d}}_h(\hat{\boldsymbol{m}})^T \hat{\boldsymbol{d}}_h(\hat{\boldsymbol{m}}) . \tag{2.13}$$

It is straightforward to verify that objective function $O(\boldsymbol{m})$, given by Eq. 2.7, and the dimensionless objective function $O_D(\hat{\boldsymbol{m}})$, given by Eq. 2.13, yield exactly the same numerical

value for $\boldsymbol{m}$ and $\hat{\boldsymbol{m}}$, respectively, provided that the vectors $\boldsymbol{m}$ and $\hat{\boldsymbol{m}}$ are related by Eq. 2.11.

The gradient of $O_D(\hat{\boldsymbol{m}})$ with respect to $\hat{\boldsymbol{m}}$ and the Hessian of $O_D(\hat{\boldsymbol{m}})$ are given, respectively, by

$$\nabla_{\hat{\boldsymbol{m}}} O_D(\hat{\boldsymbol{m}}) = \hat{\boldsymbol{m}} + G_{h,D}(\hat{\boldsymbol{m}})^T \, \hat{\boldsymbol{d}}_h(\hat{\boldsymbol{m}}) \,, \tag{2.14}$$

and

$$\mathcal{H}(\hat{\boldsymbol{m}}) = I_{N_m} + G_{h,D}(\hat{\boldsymbol{m}})^T \, G_{h,D}(\hat{\boldsymbol{m}}) \,, \tag{2.15}$$

In Eqs. 2.14 and 2.15, I_{N_m} denotes the $N_m \times N_m$ identity matrix, and $G_{h,D}(\hat{\boldsymbol{m}})$ represents the dimensionless sensitivity matrix. The dimensionless sensitivity matrix can be defined as

$$G_{h,D}(\hat{\boldsymbol{m}}) = \left[\nabla_{\hat{\boldsymbol{m}}} \left(\hat{\boldsymbol{d}}_h(\hat{\boldsymbol{m}})^T \right) \right]^T \,. \tag{2.16}$$

The dimensionless sensitivity matrix $G_{h,D}(\hat{\boldsymbol{m}})$ is related to $G_h(\boldsymbol{m})$ by [140]

$$G_{h,D}(\hat{\boldsymbol{m}}) = C_D^{-1/2} \, G_h(\boldsymbol{m}) \, C_M^{1/2} \,, \tag{2.17}$$

where the right-hand-side of Eq. 2.17 is the conventional definition of the dimensionless sensitivity matrix.

2.1.1 The Maximum a Posteriori Estimate

Until fairly recently, the history-matching problem was focused on finding the best reservoir model able to reproduce the observed data, while consistent with the prior reservoir model. This best model represents the global minimizer of the objective function $O(\boldsymbol{m})$ of Eq. 2.7, which is called the maximum a posteriori (MAP) estimate.

For virtually all practical applications, the relationship between the model $\boldsymbol{m}$ and the data predictions $\boldsymbol{d}_h$ is nonlinear. Consequently, the MAP estimate itself does not characterize the posterior pdf, and in fact, a global minimizer of $O(\boldsymbol{m})$ of Eq. 2.7 may not be unique. However, the MAP estimate plays an important role for the case of a linear relationship between the model $\boldsymbol{m}$ and the predictions $\boldsymbol{d}_h$, i.e., for the case of linear forward model, as

presented next.

For a linear function $g_h(\boldsymbol{m})$, Eq. 2.1 reduces to

$$\boldsymbol{d}_h = g_h(\boldsymbol{m}) = G_{h,L}\,\boldsymbol{m}\,, \tag{2.18}$$

where the $N_{d_h} \times N_m$ matrix $G_{h,L}$ denotes the sensitivity matrix for the linear case. As before, the entry in the ith row and jth column of the sensitivity matrix $G_{h,L}$ represents the partial derivative of the ith predicted datum (i.e., the ith entry of the data vector $\boldsymbol{d}_h$) with respect to the jth model parameter (i.e., the jth entry of the model vector $\boldsymbol{m}$). Note that $G_{h,L}$ does not depend on the model $\boldsymbol{m}$ for the linear forward model case.

With the linear relationship of Eq. 2.18, the posterior pdf of Eq. 2.6 becomes

$$\begin{aligned}\pi_L(\boldsymbol{m}) = a_d^*\, \exp\left[-\frac{1}{2}(\boldsymbol{m}-\boldsymbol{m}_{\rm pr})^T\, C_M^{-1}\,(\boldsymbol{m}-\boldsymbol{m}_{\rm pr})\right] \times \\ \exp\left[-\frac{1}{2}\Big(G_{h,L}\,\boldsymbol{m}-\boldsymbol{d}_{h,\rm obs}\Big)^T\, C_D^{-1}\,\Big(G_{h,L}\,\boldsymbol{m}-\boldsymbol{d}_{h,\rm obs}\Big)\right]\,,\end{aligned} \tag{2.19}$$

which can be rewritten as [89]

$$\pi_L(\boldsymbol{m}) = a_d^*\, \exp\left[-\frac{1}{2}(\boldsymbol{m}-\boldsymbol{m}_{\rm pos})^T\, C_{M,\rm pos}^{-1}\,(\boldsymbol{m}-\boldsymbol{m}_{\rm pos})\right]\,, \tag{2.20}$$

where

$$\boldsymbol{m}_{\rm pos} = \boldsymbol{m}_{\rm pr} - (C_M^{-1} + G_{h,L}^T\, C_D^{-1}\, G_{h,L})^{-1}\, G_{h,L}^T\, C_D^{-1}\,(G_{h,L}\,\boldsymbol{m}_{\rm pr} - \boldsymbol{d}_{h,\rm obs})\,, \tag{2.21}$$

and

$$C_{M,\rm pos} = (C_M^{-1} + G_{h,L}^T\, C_D^{-1}\, G_{h,L})^{-1}\,. \tag{2.22}$$

The posterior pdf $\pi_L(\boldsymbol{m})$ of Eq. 2.20 represents a multivariate Gaussian distribution with posterior mean $\boldsymbol{m}_{\rm pos}$ and posterior covariance matrix $C_{M,\rm pos}$. Hence, under Gaussian assumption for both the prior and measurement errors pdf, for linear relationship between the

model $\boldsymbol{m}$ and the data predictions $\boldsymbol{d}_h$, the posterior pdf for the reservoir model parameters is a Gaussian distribution. In this case, the posterior pdf is completely characterized by its mean and covariance matrix given, respectively, by Eqs. 2.21 and 2.22. Under the uncertainty quantification problem addressed here, it is theoretically easy to generate samples from a Gaussian distribution to quantify the uncertainty in future reservoir performance [26, 6, 22]. However, when $C_{M,\text{pos}}$ becomes extremely large, it is still not be computational feasible to sample $\pi_L(\boldsymbol{m})$ because sampling requires computation of a square root or Cholesky decomposition of $C_{M,\text{pos}}$ [6, 89]. It is evident from Eq. 2.20 that $\boldsymbol{m}_{\text{pos}}$ represents the global minimizer of the objective function for the linear forward model case. Therefore, the posterior mean $\boldsymbol{m}_{\text{pos}}$ also represents the MAP estimate. This is an expected result, since in the linear case, $\boldsymbol{m}_{\text{pos}}$ represents the "most probable" reservoir model.

2.1.2 The Randomized Maximum Likelihood

To generate samples from the posterior pdf, Oliver et al. [87] introduced the randomized maximum likelihood (RML). A similar approach was also independently proposed by Kitanidis [64]. To generate N_e samples from the posterior pdf $\pi(\boldsymbol{m})$ of Eq. 2.6, the RML method first samples N_e unconditioned realizations from the prior pdf. We denote each unconditioned realizations as $\boldsymbol{m}_{\text{uc},\ell}$, for $\ell = 1, 2, \ldots, N_e$, i.e., $\boldsymbol{m}_{\text{uc},\ell} \sim \mathcal{N}(\boldsymbol{m}_{\text{pr}}, C_M)$. Then, under Gaussian assumption for both the prior and measurement errors pdf, the method proceeds by solving N_e minimization problems in the form

$$\begin{aligned}\underset{\boldsymbol{m}}{\text{minimize}}\ O_\ell(\boldsymbol{m} \mid \boldsymbol{m}_{\text{uc},\ell},\, \boldsymbol{d}_{h,\text{uc},\ell}) \;=\; &+\frac{1}{2}(\boldsymbol{m}-\boldsymbol{m}_{\text{uc},\ell})^T\, C_M^{-1}\,(\boldsymbol{m}-\boldsymbol{m}_{\text{uc},\ell}) \\ &+\frac{1}{2}\Big(g_h(\boldsymbol{m})-\boldsymbol{d}_{h,\text{uc},\ell}\Big)^T\, C_D^{-1}\,\Big(g_h(\boldsymbol{m})-\boldsymbol{d}_{h,\text{uc},\ell}\Big)\,, \quad (2.23)\end{aligned}$$

for $\ell = 1, 2, \ldots, N_e$. In Eq. 2.23, the vector $\boldsymbol{d}_{h,\text{uc},\ell}$ represents a perturbation of the actual observed data $\boldsymbol{d}_{h,\text{obs}}$, which is obtained by adding a random realization $\boldsymbol{\xi}_\ell \sim \mathcal{N}(\boldsymbol{0}, C_D)$ of

the measurement errors to the actual observed data, that is

$$\boldsymbol{d}_{h,\mathrm{uc},\ell} = \boldsymbol{d}_{h,\mathrm{obs}} + \boldsymbol{\xi}_\ell \qquad \text{for} \quad \ell = 1,\, 2,\, \ldots\,,\, N_e\,, \tag{2.24}$$

consequently, $\boldsymbol{d}_{h,\mathrm{uc},\ell} \sim \mathcal{N}(\boldsymbol{d}_{h,\mathrm{obs}}, C_D)$.

For the linear forward model case, the resulting set of N_e minima of $O_\ell(\boldsymbol{m})$ in Eq. 2.23, for $\ell = 1,\, 2,\, \ldots\,,\, N_e$, represent proper samples of the posterior pdf $\pi(\boldsymbol{m})$ of Eq. 2.6 [97]. Unfortunately, for nonlinear forward model there is no guarantee that the samples generated with RML properly characterize the posterior pdf.

2.2 Gaussian Mixture Model Approximation using the Distributed Gauss-Newton Method

An approximate approach to generate samples from the posterior $\pi(\boldsymbol{m})$ of Eq. 2.6 was proposed by Gao et al. [39]. The proposed methodology proceeds by first constructing an approximation of the posterior pdf $\pi(\boldsymbol{m})$ of Eq. 2.6 using the Gaussian mixture model (GMM) probability distribution [100, 138]. Then, the authors proposed to generate an approximate set of samples from the posterior pdf by simply sampling its GMM approximation.

A GMM is a probability distribution which is formed by a weighted summation of a certain number of Gaussian distributions [100]. Hence, representing the GMM probability distribution by $\pi_{\mathrm{GMM}}(\boldsymbol{m})$, we have

$$\pi_{\mathrm{GMM}}(\boldsymbol{m}) = \sum_{\ell=1}^{N_g} w_\ell\, \mathcal{N}(\boldsymbol{m}_\ell, C_{M\ell})\,. \tag{2.25}$$

In Eq. 2.25, N_g represents the number of Gaussian distributions, $\mathcal{N}(\boldsymbol{m}_\ell, C_{M\ell})$ denotes a Gaussian distribution with mean $\boldsymbol{m}_\ell$ and covariance matrix $C_{M\ell}$, and $w_\ell > 0$, for $\ell = 1,\, 2,\, \ldots\,,\, N_g$, represents the weight of each Gaussian distribution. In order for $\pi_{\mathrm{GMM}}(\boldsymbol{m})$ of

Eq. 2.25 to be a probability distribution, the constraint

$$\sum_{\ell=1}^{N_g} w_\ell = 1 \tag{2.26}$$

must be satisfied.

Gao et al. [39] advocated that a properly constructed GMM probability distribution could satisfactorily represent the posterior pdf for practical applications. However, Rafiee [93] has shown that this statement does not hold for several applications. Gao et al. [39] suggested to build the GMM approximation centered around modes of the posterior pdf $\pi(\boldsymbol{m})$ of Eq. 2.6. Modes of a pdf represent local points of maximum probability. Therefore, the modes of the posterior pdf $\pi(\boldsymbol{m})$ of Eq. 2.6 represent minima of the objective function $O(\boldsymbol{m})$ of Eq. 2.7, or equivalently, minima of the dimensionless objective function $O_D(\hat{\boldsymbol{m}})$ of Eq. 2.13. Consequently, in the methodology proposed by Gao et al. [39], one needs to find several minima of the objective function $O_D(\hat{\boldsymbol{m}})$ of Eq. 2.13. Each different minimum of $O_D(\hat{\boldsymbol{m}})$ which is found can be used as the mean of one Gaussian distribution in the GMM approximation. Furthermore, to construct each Gaussian in the GMM approximation one must be able to determine the corresponding covariance matrix in a feasible manner. This is accomplished by using the inverse Hessian, as presented later in this dissertation.

2.2.1 The Distributed Gauss-Newton Method

To find the minima of the objective function $O_D(\hat{\boldsymbol{m}})$ of Eq. 2.13, Gao et al. [39] recommend that we use the distributed Gauss-Newton (DGN) optimization method [41]. Basically, the DGN method employs a set of reservoir models and its corresponding reservoir simulation predictions to compute an approximation of the sensitivity matrix for a given model. The approximated sensitivity matrix is used to compute both the gradient and the Hessian of the dimensionless objective function, as presented in Eqs. 2.14 and 2.15.

Using the approximate sensitivity matrix, the DGN method proceeds by solving several minimization problems, each one stating from a different selected initial guess. During

the minimization process, the original data set used to compute the approximation of the sensitivity matrix is dynamically updated. The same data set is shared among all minimization problems which are being performed. At the end of the minimization process, the minima found which yield values of the dimensionless objective function less than a given threshold are used to construct the GMM approximation of the posterior pdf. Furthermore, the covariance matrix for each Gaussian is given by the inverse Hessian of the objective function computed at each minimum found. The Hessian is calculated for each minimum using the approximate sensitivity matrix computed at the respective minimum using the final data set. Gao et al. [39] suggested the use of a trust region algorithm [40] to solve each minimization problem, as we describe latter in this section.

In an attempt to find N_e distinctive modes of $O_D(\hat{\boldsymbol{m}})$ of Eq. 2.13, the DGN method proceeds by first randomly selecting N_e initial base-cases, $\boldsymbol{m}_\ell$, for $\ell = 1, 2, \ldots, N_e$, from the prior pdf for the reservoir model parameters, i.e., the pdf $f(\boldsymbol{m})$ of Eq. 2.2. Each one of the N_e selected base-cases represent an initial guess for the N_e trust-region minimization problems which are solved simultaneously. To improve the initial approximation of the sensitivity matrix, additional reservoir models can also be randomly selected from the prior pdf. The additional reservoir models together with the N_e initial base-cases form a set of N_t reservoir models.

The DGN methods assumes that N_t is greater than the number of uncertain parameters. For the Bayesian framework adopted in this dissertation, the total number of uncertain parameters is given by N_m, i.e., the dimension of the reservoir model parameters $\boldsymbol{m}$. For practical application, N_m can potentially be equal to or larger than the number of reservoir simulation grid-blocks. To obviate this difficulty, Gao et al. [41] suggested applying the DGN method together with some dimension-reduction technique, such as the principal component analysis (PCA) [137, 110].

We discuss the PCA method in detail later in this dissertation. For now, let us assume that one is able to apply some dimension-reduction technique and represent the vector $\boldsymbol{m}$ by its reduced-dimension counterpart $\boldsymbol{z}$. We assume that the resulting dimension N_z of the

vector $\boldsymbol{z}$ is such that $N_z \ll N_m$. In this case, the corresponding objective function is given by

$$O(\boldsymbol{z}) = \frac{1}{2}(\boldsymbol{z} - \boldsymbol{z}_{\rm pr})^T \, C_Z^{-1} \, (\boldsymbol{z} - \boldsymbol{z}_{\rm pr}) + \frac{1}{2}\Big(g_h(\boldsymbol{z}) - \boldsymbol{d}_{h,\rm obs}\Big)^T \, C_D^{-1} \, \Big(g_h(\boldsymbol{z}) - \boldsymbol{d}_{h,\rm obs}\Big) \, . \tag{2.27}$$

In Eq. 2.27, $\boldsymbol{z}_{\rm pr}$ represents the prior mean of the N_z-dimensional uncertain vector $\boldsymbol{z}$, C_Z denotes the $N_z \times N_z$ prior covariance matrix of $\boldsymbol{z}$, and $g_h(\boldsymbol{z})$ represents the reservoir simulator predictions as a function of the reduced dimension vector $\boldsymbol{z}$. As we present later in this dissertation, since we assume that $\boldsymbol{m}$ follows a multivariate Gaussian distribution, by applying the PCA method the reduced dimension vector $\boldsymbol{z}$ also follows a multivariate Gaussian distribution, i.e., $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{z}_{\rm pr}, C_Z)$.

Applying similar normalization as in Eqs. 2.11 and 2.12, i.e.,

$$\hat{\boldsymbol{z}} = C_Z^{-1/2} \, (\boldsymbol{z} - \boldsymbol{z}_{\rm pr}) \, , \tag{2.28}$$

and

$$\hat{\boldsymbol{d}}_h(\hat{\boldsymbol{z}}) = C_D^{-1/2} \, \Big(g_h(\boldsymbol{z}) - \boldsymbol{d}_{h,\rm obs}\Big) \, , \tag{2.29}$$

the dimensionless objective function is given by

$$O_D(\hat{\boldsymbol{z}}) = \frac{1}{2}\hat{\boldsymbol{z}}^T \, \hat{\boldsymbol{z}} + \frac{1}{2}\hat{\boldsymbol{d}}_h(\hat{\boldsymbol{z}})^T \, \hat{\boldsymbol{d}}_h(\hat{\boldsymbol{z}}) \, . \tag{2.30}$$

In Eq. 2.30, $\hat{\boldsymbol{z}}$ denotes the dimensionless counterpart of the vector $\boldsymbol{z}$, and $\hat{\boldsymbol{d}}_h(\hat{\boldsymbol{z}})$ represents the dimensionless reservoir predictions as a function of $\hat{\boldsymbol{z}}$. The dimensionless objective function $O_D(\hat{\boldsymbol{z}})$ of Eq. 2.30, as a function of $\hat{\boldsymbol{z}}$, is the objective function that is indeed minimized to find the modes of the posterior pdf.

Following the DGN methodology, one converts all N_t models in the data set to their respective dimensionless counterparts. Hence, the initial base-cases are denoted as $\boldsymbol{z}_\ell$, for $\ell = 1, 2, \ldots, N_e$. Next, the forward model, i.e., the reservoir simulator, is run for all N_t

reservoir models to generate the reservoir predictions. The N_t reservoir models together with each respective reservoir simulation predictions form the initial data set which is used to approximate the sensitivity matrix. It is worth to stress here that using the PCA method the relation between $\boldsymbol{m}$ and $\boldsymbol{z}$ is a bijection. Therefore, although the vector $\hat{\boldsymbol{z}}$ is used in the minimization process, as explained below, when one needs to run the reservoir simulation, the unique corresponding reservoir model $\boldsymbol{m}$ is used.

For a particular reduced dimension reservoir model $\boldsymbol{z}_p$, the DGN method approximates the sensitivity matrix at $\boldsymbol{z}_p$ as follows. First, we select the N_z reservoir models $\boldsymbol{z}_c$, for $c = 1, 2, \ldots, N_z$, among the N_t reservoir models in the data set which are closest to the given model $\boldsymbol{z}_p$. The distance between any two models is computed using the Euclidean norm. With the selected N_z closest models, the approximate sensitivity matrix, $G_{h,A}(\boldsymbol{z}_p)$, for the model $\boldsymbol{z}_p$ is computed as discussed below.

For each $\boldsymbol{z}_c$, for $c = 1, 2, \ldots, N_z$, in the set of N_z closest models to $\boldsymbol{z}_p$, a first order Taylor series expansion gives

$$\boldsymbol{d}_h(\boldsymbol{z}_\ell) = \boldsymbol{d}_h(\boldsymbol{z}_p) + G_{h,A}(\boldsymbol{z}_p)\,[\boldsymbol{z}_\ell - \boldsymbol{z}_p] \qquad \text{for} \quad \ell = 1, 2, \ldots, N_z\,. \tag{2.31}$$

Letting $\Delta D^p_{d_h}$ denotes the $N_{d_h} \times N_z$ matrix whose ℓth column is given by $\boldsymbol{d}_h(\boldsymbol{z}_\ell) - \boldsymbol{d}_h(\boldsymbol{z}_p)$, and ΔZ^p denotes the $N_z \times N_z$ matrix whose ℓth column is given by $\boldsymbol{z}_\ell - \boldsymbol{z}_p$, the linear system given in Eq. 2.31 can be rewritten as

$$\Delta D^p_{d_h} = G_{h,A}(\boldsymbol{z}_p)\,\Delta Z^p\,, \tag{2.32}$$

or equivalently

$$(\Delta Z^p)^T\,(G_{h,A}(\boldsymbol{z}_p))^T = \left(\Delta D^p_{d_h}\right)^T\,, \tag{2.33}$$

which can be solved for the columns of $(G_{h,A}(\boldsymbol{z}_p))^T$ in order to find $G_{h,A}(\boldsymbol{z}_p)$. Note this assumes that the $N_z \times N_z$ matrix ΔZ^p is non-singular, which is only the case if no two of the selected models $\boldsymbol{z}_\ell$ are identical. However, to attempt to ensure that the matrix $(\Delta Z^p)^T$ is

not ill-conditioned, we would make sure that the $\boldsymbol{z}_\ell$'s chosen to compute $G_{h,A}(\boldsymbol{z}_p)$ are not too close. Note the entry in ith row, for $i = 1, 2, \cdots, N_{d_h}$, and ℓth column, for $\ell = 1, 2, \cdots, N_z$, of matrix $\Delta D^p_{d_h}$ is given by $d_{h,i}(\boldsymbol{z}_\ell) - d_{h,i}(\boldsymbol{z}_p)$, where $d_{h,i}$ denotes the ith entry component, for $i = 1, 2, \cdots, N_{d_h}$, of the column vector of predicted data, $\boldsymbol{d}_h$. Similarly, the entry in jth row, for $j = 1, 2, \cdots, N_z$, and ℓth column, for $\ell = 1, 2, \cdots, N_z$, of matrix ΔZ^p is given by $z_{\ell,j} - z_{p,j}$, where $z_{\ell,j}$ is the jth entry of the column vector $\boldsymbol{z}_\ell$, and $z_{p,j}$ is the jth entry of the column vector $\boldsymbol{z}_p$.

Gao et al. [41] adopted a trust region minimization algorithm to minimize $O_D(\hat{\boldsymbol{z}})$ of Eq. 2.30. The trust region algorithm is presented in Appendix B. At each iteration of each N_e minimization problem conducted, the trust region algorithm search for minima of the objective function $O_D(\hat{\boldsymbol{z}})$ of Eq. 2.30 by constructing a local quadratic approximation of $O_D(\hat{\boldsymbol{z}})$ around the current dimensionless updated base-case. Denoting the quadratic approximation for the ℓth minimization problem, for $\ell = 1, 2, \ldots, N_e$, at the nth iteration, as $q^{(n)}_\ell(\delta\hat{\boldsymbol{z}})$, one gets

$$q^{(n)}_\ell(\delta\hat{\boldsymbol{z}}) = O_D(\hat{\boldsymbol{z}}^{(n)}_\ell) + \nabla_{\hat{\boldsymbol{z}}} O_D(\hat{\boldsymbol{z}}^{(n)}_\ell)^T\, \delta\hat{\boldsymbol{z}} \,+\, \frac{1}{2}\delta\hat{\boldsymbol{z}}^T\, \mathcal{H}(\hat{\boldsymbol{z}}^{(n)}_\ell)\, \delta\hat{\boldsymbol{z}}\,. \tag{2.34}$$

In Eq. 2.34, $\delta\hat{\boldsymbol{z}}$ represents the search direction, $\hat{\boldsymbol{z}}^{(n)}_\ell$ denotes the current estimate of $\hat{\boldsymbol{z}}$ of the ℓth minimization problem at the nth iteration, which is the current estimate of the minimum of $O_D(\hat{\boldsymbol{z}})$ of Eq. 2.30 for the ℓth minimization problem. As discussed earlier, $\hat{\boldsymbol{z}}^{(n)}_\ell$ is the dimensionless counterpart of $\boldsymbol{z}^{(n)}_\ell$, where these two vectors are related by

$$\hat{\boldsymbol{z}}^{(n)}_\ell = C_Z^{-1/2}\,(\boldsymbol{z}^{(n)}_\ell - \boldsymbol{z}_{\rm pr}) \qquad \Longleftrightarrow \qquad \boldsymbol{z}^{(n)}_\ell = C_Z^{1/2}\,\hat{\boldsymbol{z}}^{(n)}_\ell + \boldsymbol{z}_{\rm pr}\,. \tag{2.35}$$

In this context, $\boldsymbol{z}^{(n)}_\ell$ represents the current estimate of the minimum of the minimum of $O(\boldsymbol{z})$ of Eq. 2.27 for the ℓth minimization problem at the nth iteration. Notice that for $n = 0$, the required $\boldsymbol{z}^{(0)}_\ell$, for $\ell = 1, 2, \ldots, N_e$, are given by the initial selected base-cases.

Also in Eq. 2.34, $O_D(\hat{\boldsymbol{z}}^{(n)}_\ell)$ is computed using Eq. 2.30 for $\hat{\boldsymbol{z}} = \hat{\boldsymbol{z}}^{(n)}_\ell$. Furthermore, the required gradient, $\nabla_{\hat{\boldsymbol{z}}} O_D(\hat{\boldsymbol{z}})$, and Hessian, $\mathcal{H}(\hat{\boldsymbol{z}})$, of $O_D(\hat{\boldsymbol{z}})$ of Eq. 2.30 are given, respectively,

by

$$\nabla_{\hat{\boldsymbol{z}}} O_D(\hat{\boldsymbol{z}}) = \hat{\boldsymbol{z}} + G_{h,D}(\hat{\boldsymbol{z}})^T \, \hat{\boldsymbol{d}}_h(\hat{\boldsymbol{z}}) \,, \tag{2.36}$$

and

$$\mathcal{H}(\hat{\boldsymbol{z}}) = I_{N_z} + G_{h,D}(\hat{\boldsymbol{z}})^T \, G_{h,D}(\hat{\boldsymbol{z}}) \,, \tag{2.37}$$

In Eqs. 2.36 and 2.37, I_{N_z} denotes the $N_z \times N_z$ identity matrix, and $G_{h,D}(\hat{\boldsymbol{z}})$ represents the dimensionless $N_{d_h} \times N_z$ sensitivity matrix computed by

$$G_{h,D}(\hat{\boldsymbol{z}}) = C_D^{-1/2} \, G_h(\boldsymbol{z}) \, C_Z^{1/2} \,, \tag{2.38}$$

where $G_h(\boldsymbol{z})$ denotes the $N_{d_h} \times N_z$ sensitivity matrix given by

$$G_h(\boldsymbol{z}) = \left[\nabla_{\boldsymbol{z}} \Big(g_h(\boldsymbol{z})^T \Big) \right]^T \,. \tag{2.39}$$

In the DGN method, the gradient, $\nabla_{\hat{\boldsymbol{z}}} O_D(\hat{\boldsymbol{z}}_\ell^{(n)})$, and Hessian, $\mathcal{H}(\hat{\boldsymbol{z}}_\ell^{(n)})$, required in Eq. 2.34 are computed, respectively, using Eqs. 2.36 and 2.37 for $\hat{\boldsymbol{z}} = \hat{\boldsymbol{z}}_\ell^{(n)}$, with the approximate sensitivity matrix $G_{h,A}(\boldsymbol{z}_p)$, computed by solving the system of Eq. 2.33 for $\boldsymbol{z}_p = \boldsymbol{z}_\ell^{(n)}$, as a replacement of the sensitivity matrix $G_h(\boldsymbol{z})$ in Eq. 2.38.

To apply the complete procedure with the equations described in Appendix B, one should set $\boldsymbol{x}_p^{(n)} = \hat{\boldsymbol{z}}_\ell^{(n)}$, $\delta \boldsymbol{x} = \delta \hat{\boldsymbol{z}}$, $O(\boldsymbol{x}_p^{(n)}) = O_D(\hat{\boldsymbol{z}}_\ell^{(n)})$, $\mathcal{G}(\boldsymbol{x}_p^{(n)}) = \nabla_{\hat{\boldsymbol{z}}} O_D(\hat{\boldsymbol{z}}_\ell^{(n)})$ and $\mathcal{H}(\boldsymbol{x}_p^{(n)}) = \mathcal{H}(\hat{\boldsymbol{z}}_\ell^{(n)})$, for $\ell = 1, 2, \ldots, N_e$, so Eqs. 2.34 and B.5 exactly correspond.

As described in Appendix B, when the quadratic approximation $q_\ell^{(n)}(\delta \hat{\boldsymbol{z}})$ of Eq. 2.34 represents a reasonable approximation of $O_D(\hat{\boldsymbol{z}})$ of Eq. 2.30, the updated model replaces the current dimensionless base-case $\hat{\boldsymbol{z}}_\ell^{(n)}$ (see Algorithm B.1), as described next. Notice that to assess the quality of the quadratic approximation, one needs to run the reservoir simulator to compute the predictions for the updated model. For the ℓth minimization problem at the nth iteration, the current dimensionless base-case, which represents the current estimate of

the minimum of $O_D(\hat{\boldsymbol{z}})$ of Eq. 2.30, is updated by

$$\hat{\boldsymbol{z}}_\ell^{(n+1)} = \hat{\boldsymbol{z}}_\ell^{(n)} + \delta\hat{\boldsymbol{z}}^* \qquad \text{for} \quad \ell = 1,\, 2,\, \ldots\,,\, N_e\,. \tag{2.40}$$

In Eq. 2.40, $\delta\hat{\boldsymbol{z}}^*$ represents the solution of the trust region sub-problem (see Eqs. B.5 and B.6).

In the DGN methodology, the data set used to evaluate the approximation of the sensitivity matrix is dynamically updated by the incorporation of the updated base-case $\boldsymbol{z}_\ell^{(n+1)}$ at each iteration of each minimization problem. $\boldsymbol{z}_\ell^{(n+1)}$ is computed from $\hat{\boldsymbol{z}}_\ell^{(n+1)}$ using Eq. 2.35. However, Gao et al. [41] suggested that one first verify that the distance of the updated base-case tor the models already in the data set is not too small. Specifically, they compute

$$d_{\ell,\min}^{(n)} = \underset{\boldsymbol{z}_t \,\in\, \text{Data Set}}{\arg\min} \left\| \boldsymbol{z}_\ell^{(n+1)} - \boldsymbol{z}_t \right\|_2 , \tag{2.41}$$

where $\boldsymbol{z}_t$, for $t = 1,\, 2,\, \ldots\,,\, N_t$, represents the models in the current data set, and $\|\cdot\|$ denotes the Euclidean norm. The model $\boldsymbol{z}_\ell^{(n+1)}$ is added to the current data set whenever $d_{\ell,\min}^{(n)} > d_{\min}$, where the parameter $d_{\min}$ is a minimum distance specified by the user. Gao et al. [41] indicated that at each iteration, each updated model in the data set become closer to one of the minima of $O_D(\hat{\boldsymbol{z}})$ of Eq. 2.30, consequently, the approximate sensitivity matrix becomes more accurate as the minimization process proceeds. It is important to emphasize that in order to incorporate the model $\boldsymbol{z}_\ell^{(n+1)}$ into the current data set, one needs to run the reservoir simulation to compute the corresponding predictions. However, these predictions were already computed when assessing the quadratic approximation, thus no extra reservoir simulation is required to add the updated model to the data set.

Gao et al. [41] suggested that we regard a particular minimization problem as converged whenever no improvement of the estimate of the minimum is observed or the updated trust region radius becomes smaller than a given threshold $\delta_{\min}$ (see Algorithm B.1). The parameter $\delta_{\min}$ is defined by the user. Following the procedure presented in Appendix B, we

evaluate the improvement of the estimate of the minimum of $O_D(\hat{\boldsymbol{z}})$ of Eq. 2.30 as

$$\rho_{\ell,\text{stop}}^{(n)} \equiv \frac{\text{abs}\left[O_D(\hat{\boldsymbol{z}}_\ell^{(n+1)}) - O_D(\hat{\boldsymbol{z}}_\ell^{(n)})\right]}{\text{abs}\left[O_D(\hat{\boldsymbol{z}}_\ell^{(n+1)})\right]} \,. \tag{2.42}$$

We consider that no improvement is obtained whenever $\rho_{\ell,\text{stop}}^{(n)} \leq \epsilon_{\min}$, where $\epsilon_{\min}$ is an accuracy parameter defined by the user. A numerical value in the order of $\epsilon_{\min} = 1.0\text{E} - 6$ seems to suffice for practical problems. The DGN method is regarded as converged when all minimization problems converge. Algorithm 2.1 summarizes the DGN method.

2.2.2 The Gaussian Mixture Model Approximation of the Posterior

Using the DGN method one can find up to N_e local minima of the objective function $O_D(\hat{\boldsymbol{z}})$ of Eq. 2.30, or equivalently, find N_e local minima of the objective function $O(\boldsymbol{z})$ of Eq. 2.27. Gao et al. [39] recommended that one construct the GMM approximation of the posterior pdf using only those minima found which result in an objective function value less than a given threshold. Let $\boldsymbol{z}_s$, for $s = 1, 2, \ldots, N_s$, represent the N_s selected modes of the posterior among the N_e minima found. Consequently, the values of $O(\boldsymbol{z}_s)$ computed using Eq. 2.27 with $\boldsymbol{z} = \boldsymbol{z}_s$, for $s = 1, 2, \ldots, N_s$, have numerical value less than a given threshold. Furthermore, Gao et al. [39] suggested separating the N_s selected minima into distinct clusters. By clustering the modes found of the posterior pdf, one avoids having essentially the same mode represented by more than one Gaussian in the GMM approximation.

The procedure suggested by Gao et al. [39] to cluster the N_s minima found was to first compute the approximated sensitivity matrix $G_{h,A}(\boldsymbol{z}_s)$ for each selected mode. Therefore, for a particular mode the approximated sensitivity matrix is computed using Eq. 2.33 and the N_z closest models to the mode, which are selected among the models in the final updated data set which was obtained at the end of the minimization process. For two different selected

Algorithm 2.1: The DGN Method

1. Select N_e base-cases $\boldsymbol{m}_\ell$, for $\ell = 1, 2, \ldots, N_e$, by sampling the prior pdf for $\boldsymbol{m}$. If required/desired, sample additional models to obtain a total of N_t reservoir models.

2. Apply a dimension-reduction technique to transform the N_t reservoir models to their reduced-order counterparts $\boldsymbol{z}$'s. The base-cases become $\boldsymbol{z}_\ell$, for $\ell = 1, 2, \ldots, N_e$.

3. Run the reservoir simulator for all N_t reservoir models and assemble the initial data set.

4. Define the trust region parameters η_1, η_2 and $\delta_{\max}$ (see Algorithm B.1), the minimum distance for updating the data set, $d_{\min}$, and the convergence criteria $\epsilon_{\min}$ and $\delta_{\min}$.

5. Set $n = 0$, $r_{\text{conv}} = 0$, and $\boldsymbol{z}_\ell^{(n)} = \boldsymbol{z}_\ell$, for $\ell = 1, 2, \ldots, N_e$.

6. **While** ($r_{\text{conv}} < N_e$)

   - **For** ($\ell = 1$ to $N_e - r_{\text{conv}}$)
     - Select the N_z models in the data set which are closest to $\boldsymbol{z}_\ell^{(n)}$ and compute the approximate sensitivity matrix $G_{h,A}(\boldsymbol{z}_\ell^{(n)})$ by solving Eq. 2.33 with $\boldsymbol{z}_p = \boldsymbol{z}_\ell^{(n)}$.
     - Compute $\hat{\boldsymbol{z}}_\ell^{(n)}$ by using Eq. 2.35. Compute $G_{h,D}(\hat{\boldsymbol{z}}_\ell^{(n)})$ by using Eq. 2.38 for $G_h(\boldsymbol{z}) = G_{h,A}(\boldsymbol{z}_\ell^{(n)})$. Compute $O_D(\hat{\boldsymbol{z}}_\ell^{(n)})$, $\nabla_{\hat{\boldsymbol{z}}} O_D(\hat{\boldsymbol{z}}_\ell^{(n)})$, and $\mathcal{H}(\hat{\boldsymbol{z}}_\ell^{(n)})$ by using, respectively, Eqs. 2.30, 2.36 and 2.37 for $\hat{\boldsymbol{z}} = \hat{\boldsymbol{z}}_\ell^{(n)}$.
     - Assemble the quadratic approximation $q_\ell^{(n)}(\delta\hat{\boldsymbol{z}})$ of Eq. 2.34, and solve ONE iteration of the trust region minimization algorithm to compute $\hat{\boldsymbol{z}}_\ell^{(n+1)}$ and the updated trust region radius $\delta_\ell^{(n)}$ (see Algorithm B.2). Compute $\rho_{\text{stop}}^{(n)}$ by using Eq. 2.42.
     - Compute $d_{\ell,\min}^{(n)}$ by using Eq. 2.41. If $d_{\ell,\min}^{(n)} > d_{\min}$, ADD $\boldsymbol{z}_\ell^{(n+1)}$ to the data set. Set $N_t = N_t + 1$.
     - **If** ($\rho_{\text{stop}}^{(n)} \le \epsilon_{\min}$ or $\delta_{\text{p}}^{(n)} \le \delta_{\min}$)
       - Set $r_{\text{conv}} = r_{\text{conv}} + 1$.
       - REMOVE the ℓth minimization problem from the minimization loop, and reorder the remaining (non-converged) minimization problems from 1 to $N_e - r_{\text{conv}}$.

       End If

     End For

   - Set $n = n + 1$

minima $\boldsymbol{z}_{s_i}$ and $\boldsymbol{z}_{s_j}$, for $1 \leq s_i \neq s_j \leq N_s$, whenever

$$\left\| G_{h,A}(\boldsymbol{z}_{s_i}) - G_{h,A}(\boldsymbol{z}_{s_j}) \right\|_F < \delta_F \,, \tag{2.43}$$

and

$$\left\| \boldsymbol{z}_{s_i} - \boldsymbol{z}_{s_j} \right\| < \delta_E \,, \tag{2.44}$$

one puts $\boldsymbol{z}_{s_i}$ and $\boldsymbol{z}_{s_j}$ in the same cluster. Otherwise, $\boldsymbol{z}_{si}$ and $\boldsymbol{z}_{sj}$ belong to two distinct clusters. In Eqs. 2.43 and 2.44, $\| \cdot \|_F$ represents the Frobenius norm [45], and δ_F and δ_E represents two user defined precision parameters.

By comparing the N_s selected minima, one determines the total number of clusters, which is denoted by N_c. Then, Gao et al. [39] recommend averaging the modes and approximate sensitivity matrices within each distinct cluster, i.e.,

$$\overline{\boldsymbol{z}}_c = \frac{1}{N_{c_t}} \sum_{\nu=1}^{N_{c_t}} \boldsymbol{z}_{s,c_\nu} \qquad \text{for} \quad c = 1,\, 2,\, \ldots\,,\, N_c \,, \tag{2.45}$$

and

$$G_{h,A}(\overline{\boldsymbol{z}}_c) = \frac{1}{N_{c_t}} \sum_{\nu=1}^{N_{c_t}} G_{h,A}(\boldsymbol{z}_{s,c_\nu}) \qquad \text{for} \quad c = 1,\, 2,\, \ldots\,,\, N_c \,. \tag{2.46}$$

In Eqs. 2.45 and 2.46, $\overline{\boldsymbol{z}}_c$ and $G_{h,A}(\overline{\boldsymbol{z}}_c)$, for $c = 1,\, 2,\, \ldots\,,\, N_c$, represents, respectively, the average mode and average sensitivity matrix for the cth cluster, N_{c_t} represents the total number of modes which belongs to the cth cluster, and $\boldsymbol{z}_{s,c_\nu}$ represents the νth mode belonging to the cth cluster.

The GMM approximation of the posterior pdf is then constructed using all clusters found, which results in

$$\pi_{\text{GMM}}(\boldsymbol{z}) = \sum_{c=1}^{N_c} w_c \, \mathcal{N}(\overline{\boldsymbol{z}}_c, \overline{C}_{zc}) \,. \tag{2.47}$$

In Eq. 2.47, w_c must satisfy the condition given by Eq. 2.26, each Gaussian's mean $\overline{\boldsymbol{z}}_c$, for $c = 1,\, 2,\, \ldots\,,\, N_c$, is given by Eq. 2.45, and each Gaussian's $N_z \times N_z$ covariance matrix $\overline{C}_{zc}$,

for $c = 1, 2, \ldots, N_c$, is computed as

$$\overline{C}_{zc} = \left[\mathcal{H}(\overline{\boldsymbol{z}}_c)\right]^{-1} = \left[C_Z^{-1} + G_{h,A}(\overline{\boldsymbol{z}}_c)^T\, C_D^{-1}\, G_{h,A}(\overline{\boldsymbol{z}}_c)\right]^{-1}, \tag{2.48}$$

that is, $\overline{C}_{zc}$ is given by the inverse of the Hessian $\mathcal{H}(\overline{\boldsymbol{z}}_c)$ of the objective function $O(\boldsymbol{z})$ of Eq. 2.27, computed at $\boldsymbol{z} = \overline{\boldsymbol{z}}_c$, for $c = 1, 2, \ldots, N_c$.

The pdf $\pi_{\text{GMM}}(\boldsymbol{z})$ of Eq. 2.47 is the GMM probability distribution used by Gao et al. [39] to approximate the posterior pdf for reservoir model parameters. Gao et al. [39] suggested that one simply needs to sample this GMM to quantify the uncertainty in reservoir predictions.

2.3 Sampling the Posterior Probability Distribution Function using Markov Chain Monte Carlo

A rigorous technique to sample any given pdf is the Markov chain Monte Carlo (MCMC) method. In this research, we adopt the Metropolis-Hastings [76, 56] MCMC algorithm to sample the posterior pdf $\pi(\boldsymbol{m})$ of Eq. 2.6. The Metropolis-Hastings algorithm is given in Algorithm 2.2. As discussed earlier, starting from any initial plausible state, the Metropolis-Hastings algorithm is guaranteed to converge [125]. Hence, after the burn-in period ends, subsequently generated states represent samples from the target distribution. A prominent characteristic of the Metropolis-Hastings algorithm is that the Metropolis-Hastings acceptance probability of Eq. 2.49 only depends on ratios of the target pdf. Consequently, one only needs to know the target pdf up to a normalizing constant. This is an important feature for petroleum reservoir applications, since it is computationally infeasible to determine the normalizing constant a_d^* of the posterior pdf $\pi(\boldsymbol{m})$ of Eq. 2.6.

As one can see in Algorithm 2.2, the Metropolis-Hastings algorithm requires the selection of a proposal distribution. The proposal distribution is used for both proposing a new candidate state in the chain and evaluating the Metropolis-Hastings acceptance probability of Eq. 2.49. It is widely recognized that the computational efficiency of the the Metropolis-

Algorithm 2.2: Metropolis-Hastings MCMC algorithm

1. Select the proposal distribution, $q(\widetilde{\boldsymbol{m}}_{s+1}|\,\boldsymbol{m}_s)$, which gives the probability of proposing a candidate for the new state in the chain, $\widetilde{\boldsymbol{m}}_{s+1}$, given the current state $\boldsymbol{m}_s$.

2. Set the state counter $s = 0$ and choose the initial state, $\boldsymbol{m}_0$, in the chain.

3. From the current state, $\boldsymbol{m}_s$, propose a new candidate state in the chain, $\widetilde{\boldsymbol{m}}_{s+1}$, by sampling the proposal distribution.

4. Evaluate the Metropolis-Hastings acceptance probability, $\alpha(\boldsymbol{m}_s,\ \widetilde{\boldsymbol{m}}_{s+1})$, which is given by
$$\alpha(\boldsymbol{m}_s,\ \widetilde{\boldsymbol{m}}_{s+1}) = \min\left\{1,\ \frac{\pi(\widetilde{\boldsymbol{m}}_{s+1})\ q(\boldsymbol{m}_s|\ \widetilde{\boldsymbol{m}}_{s+1})}{\pi(\boldsymbol{m}_s)\ q(\widetilde{\boldsymbol{m}}_{s+1}|\,\boldsymbol{m}_s)}\right\}. \tag{2.49}$$

5. Generate a random number u_s from the uniform distribution $U[0,1]$.

6. If $u_s < \alpha(\boldsymbol{m}_s,\ \widetilde{\boldsymbol{m}}_{s+1})$, the proposed candidate is accepted as the new state in the chain, i.e., set
$$\boldsymbol{m}_{s+1} = \widetilde{\boldsymbol{m}}_{s+1}\,.$$
If $u_s \geq \alpha(\boldsymbol{m}_s,\ \widetilde{\boldsymbol{m}}_{s+1})$, the proposed candidate is rejected and the current state is repeated in the chain, i.e., set
$$\boldsymbol{m}_{s+1} = \boldsymbol{m}_s\,.$$

7. Increase the chain counter by making $s = s + 1$ and return to Step 3 until the desired chain length is achieved.

Hastings algorithm, i.e., its convergence rate, greatly depends on the particular choice of the proposal distribution. In fact, Metropolis et al. [76] recognized that the performance of their newly proposed algorithm was greatly dependent on the particular tuning choice of their proposal distribution parameters. For practical applications, the burn-in period can also represent a challenge. An excessively long burn-in period can render the method infeasible for cases in which the evaluation of the target pdf is computationally expensive, as is the case for the posterior pdf consider in this dissertation. A judicious selection of the proposal distribution can potentially reduce the burn-in period, which can significantly increase the computational efficiency of the Metropolis-Hastings algorithm.

2.3.1 The Random Walk Metropolis-Hastings Algorithm

Many variants of the Metropolis-Hastings algorithm adopt a Gaussian proposal distribution, $q_{\rm gp}(\widetilde{\boldsymbol{m}}_{s+1}|\,\boldsymbol{m}_s)$, which is centered at the current state $\boldsymbol{m}_s$, i.e.,

$$q_{\rm gp}(\widetilde{\boldsymbol{m}}_{s+1}|\,\boldsymbol{m}_s) = \frac{1}{\sqrt{(2\pi)^{N_m}\,|C_{\rm gp}|}}\,\exp\left[-\frac{1}{2}(\widetilde{\boldsymbol{m}}_{s+1}-\boldsymbol{m}_s)^T\,C_{\rm gp}^{-1}\,(\widetilde{\boldsymbol{m}}_{s+1}-\boldsymbol{m}_s)\right]. \qquad (2.50)$$

This approach is normally referred to as the random walk Metropolis-Hastings algorithm.

The covariance matrix $C_{\rm gp}$ in the pdf of Eq. 2.50 has to be defined beforehand. When a prior pdf is available, normally a scaled prior covariance matrix is adopted in the proposal distribution $q_{\rm gp}(\widetilde{\boldsymbol{m}}_{s+1}|\,\boldsymbol{m}_s)$ of Eq. 2.50, i.e., $C_{\rm gp} = \sigma^2\,C_M$, for some scaling factor σ. When no reasonable prior covariance matrix is known, usually $C_{\rm gp}$ is set equal to the identity matrix multiplied by some scaling factor, i.e.,

$$C_{\rm gp} = \sigma^2\,I_{N_m}\,. \qquad (2.51)$$

In Eq. 2.51, the scaling factor is represented by σ, and I_{N_m} denotes the N_m-dimensional identity matrix. In this case, the efficiency of the Metropolis-Hastings algorithm depends on the particular choice of the scaling factor σ. It is well recognized that using a scaling factor that is small results in a high acceptance rate. In this context, the acceptance rate denotes the ratio between the number of accepted new states in the chain and the total number of states proposed, i.e., the current length of the Markov chain. Although a small scaling factor will give a high acceptance rate, the resulting Markov chain will not be well mixed. The main reason is that for a small scaling factor the proposed new candidate states in the chain will potentially be close to the current state. Consequently, it will take an excessive large number of states in the chain to properly investigate and characterize the target pdf. Conversely, using a too large value of the scaling factor tends to result in a small acceptance rate. Basically, with a large scaling factor the proposed candidates to new state in the chain tend to be far from the current state, which could represent regions of low probability. Therefore, the resulting Markov chain often tends to remain trapped at the same state for a long period, i.e., for many iterations. Hence, again an excessive large number of states may

be required to properly characterize the target pdf.

For a multivariate target pdf in which the components of the random vector are independent, Roberts et al. [102] and Gelman et al. [43] showed that the optimal acceptance rate, for the random walk Metropolis-Hastings algorithm, approaches the numerical value of 0.234 as the dimension of the multivariate random vector approaches infinity. The numerical examples presented in the latter work showed that an acceptance rate of approximately 0.234 enables good performance of the random walk Metropolis-Hastings algorithm for dimensions of the random vector as low as 6. Although the proof of Roberts et al. [102] and Gelman et al. [43] only applies for a restrictive condition, some numerical results suggest that an acceptance rate of approximately 0.234 is close to optimal for more general conditions [106]. Nevertheless, Roberts and Tweedie [104] and Roberts and Rosenthal [101] suggested that an efficient random walk Metropolis-Hastings algorithm can be accomplished by using an acceptance rate between 0.20 and 0.50. Following this recommendation, we chose to tune our proposed algorithm to deliver an acceptance rate between 0.20 and 0.50 to achieve good performance, as we discuss later in this dissertation.

2.4 Two-Level Markov Chain Monte Carlo Metropolis-Hastings Algorithm

For a multi-modal target pdf, as is likely the case for petroleum reservoir applications, the selection and tuning of the proposal distribution is a challenging assignment. However, a close examination of the Metropolis-Hastings acceptance probability of Eq. 2.49 reveals that if one was able to propose states directly from the target pdf, any proposed state would be accepted with probability one. Based on this observation, it is widely accepted that a choice of proposal distribution which is close to the target pdf results in an efficient MCMC Metropolis-Hastings sampling algorithm. Furthermore, with such a proposal distribution one expects a reduction in the burn-in period, resulting in a more computational affordable algorithm for practical cases, for which applying the forward model is computationally expensive. This idea motivated the work of Oliver et al. [88, 12], however, their efforts to build a good proposal distribution were not highly successful.

Guided by this idea and motivated by the work of Gao et al. [39], Li and Reynolds [69, 68] proposed an efficient two-level MCMC Metropolis-Hastings sampling algorithm. The conceptual main difference between Li and Reynolds [69, 68] developments, and the original idea of Gao et al. [39] is that Li and Reynolds suggested to construct the GMM approximation of the posterior pdf to use as a proposal distribution in a MCMC framework, while Gao et al. advocated directly sampling the GMM approximation. The methodology proposed by Li and Reynolds [69, 68] consists of two steps. In the first step, one constructs the GMM approximation of the posterior pdf. Then, in the second step, one samples the posterior pdf with the MCMC Metropolis-Hastings algorithm and the GMM approximation as the proposal distribution, as we discuss in the next subsection.

2.4.1 Two-Level Markov Chain Monte Carlo using a Reservoir Simulator with Adjoint Capability

Similar to Gao et al. [39], Li and Reynolds [69, 68] suggested the construction of a GMM approximation of the posterior pdf $\pi(\boldsymbol{m})$ of Eq. 2.6. In the Li and Reynolds [69, 68] approach, one starts from N_e distinct initial reservoir models. Then, one solves N_e minimization problems to find minima of the objective function $O(\boldsymbol{m})$ of Eq. 2.7, in an attempt to find modes of the posterior pdf. The N_e initial reservoir models are samples from the prior pdf. Each initial reservoir model represents the initial guess in each one of the N_e minimization problems conducted. Li and Reynolds [69, 68] adopted a gradient based optimization method to solve the minimization problems. The gradient and Hessian of the objective function are given, respectively, by Eqs. 2.8 and 2.9. To compute the required sensitivity matrix of Eq. 2.10, the authors adopted an in-house reservoir simulator with adjoint capability.

At the end of the minimization process, Li and Reynolds [69, 68] suggested keeping only the modes which result in an objective function values less than a given threshold. The authors recommended clustering the selected modes into N_c clusters using the k-medoids clustering algorithm [63]. Using the N_c clusters, Li and Reynolds [69, 68] construct a GMM

approximation of the posterior pdf as

$$\pi_{\text{GMM}}(\boldsymbol{m}) \equiv \sum_{\ell=1}^{N_c} w_\ell \, \mathcal{N}(\boldsymbol{m}_\ell^*, C_{M\ell}^*)$$
$$= \sum_{\ell=1}^{N_c} \frac{w_\ell}{\sqrt{(2\pi)^{N_m} \, |C_{M\ell}^*|}} \exp\left[-\frac{1}{2}(\boldsymbol{m} - \boldsymbol{m}_\ell^*)^T \, C_{M\ell}^{*-1} \, (\boldsymbol{m} - \boldsymbol{m}_\ell^*)\right] . \tag{2.52}$$

In Eq. 2.52, $w_\ell > 0$, for $\ell = 1, 2, \ldots, N_c$, represents a positive weight which satisfies the constraint in Eq. 2.26, $\boldsymbol{m}_\ell^*$ represents the ℓth cluster, which is used as the mean of the ℓth Gaussian, and the matrix $C_{M\ell}^*$ is given by the inverse of the objective function Hessian evaluated at $\boldsymbol{m}_\ell^*$. For each Gaussian in the GMM approximation, the required Hessian (see Eq. 2.9) is calculated using the sensitivity matrix computed with the adjoint method at the respective cluster $\boldsymbol{m}_\ell^*$, for $\ell = 1, 2, \ldots, N_c$.

In the second step of the Li and Reynolds [69, 68] approach, one samples the posterior pdf using the Metropolis-Hastings algorithm presented in Algorithm 2.3. As one can see, the proposal distribution $q_{\text{LR}}(\widetilde{\boldsymbol{m}}_{s+1} | \, \boldsymbol{m}_s)$ of Eq. 2.53 is independent of the current state $\boldsymbol{m}_s$. This is a prominent characteristic of the proposal distribution introduced by Li and Reynolds [69, 68]. Importantly, it promote good mixing of the resulting Markov chain [69]. Using the $q_{\text{LR}}(\widetilde{\boldsymbol{m}}_{s+1} | \, \boldsymbol{m}_s)$ of Eq. 2.53 the proposed candidates for the new state in the chain tend to come from different regions of the target pdf, which results in a better investigation of the posterior pdf. Furthermore, since the proposal distribution is constructed around the modes of the posterior pdf, for cases where the modes of the posterior pdf are separated by regions of low probability, the independent proposal distribution of Eq. 2.53 mitigates the problem of having the Markov chain trapped at the same mode for a large number of iterations. For reservoir petroleum applications, this characteristic of the proposal distribution of Li and Reynolds results in considerable reduction in the duration of the burn-in period [69, 68].

Monitoring the Convergence of the Markov Chain:

As discussed earlier in this dissertation, under reasonable conditions, starting from any randomly selected initial sate, the Metropolis-Hastings algorithm is guaranteed to con-

Algorithm 2.3: Li and Reynolds [69, 68] Metropolis-Hastings MCMC algorithm

1. Construct the GMM proposal distribution using the clusters found in the first step

$$q_{\text{LR}}(\widetilde{\boldsymbol{m}}_{s+1}|\,\boldsymbol{m}_s) = \sum_{\ell=1}^{N_c} \frac{w_\ell}{\sqrt{(2\pi)^{N_m}\,|C^*_{M\ell}|}}\,\exp\left[-\frac{1}{2}(\widetilde{\boldsymbol{m}}_{s+1}-\boldsymbol{m}^*_\ell)^T\,C^{*-1}_{M\ell}\,(\widetilde{\boldsymbol{m}}_{s+1}-\boldsymbol{m}^*_\ell)\right]. \tag{2.53}$$

   The proposal distribution $q_{\text{LR}}(\widetilde{\boldsymbol{m}}_{s+1}|\,\boldsymbol{m}_s)$ gives the probability of proposing a new candidate state in the chain, $\widetilde{\boldsymbol{m}}_{s+1}$, given the current state $\boldsymbol{m}_s$.

2. Set the state counter $s = 0$ and choose the initial state, $\boldsymbol{m}_0$, in the chain by sampling the proposal distribution $q_{\text{LR}}(\widetilde{\boldsymbol{m}}_{s+1}|\,\boldsymbol{m}_s)$ of Eq. 2.53.

3. From the current state, $\boldsymbol{m}_s$, propose a new candidate state in the chain, $\widetilde{\boldsymbol{m}}_{s+1}$, by sampling the proposal distribution $q_{\text{LR}}(\widetilde{\boldsymbol{m}}_{s+1}|\,\boldsymbol{m}_s)$ of Eq. 2.53.

4. Evaluate the Metropolis-Hastings acceptance probability, $\alpha(\boldsymbol{m}_s,\,\widetilde{\boldsymbol{m}}_{s+1})$, given by

$$\alpha(\boldsymbol{m}_s,\,\widetilde{\boldsymbol{m}}_{s+1}) = \min\left\{1,\,\frac{\pi(\widetilde{\boldsymbol{m}}_{s+1})\;q_{\text{LR}}(\boldsymbol{m}_s|\,\widetilde{\boldsymbol{m}}_{s+1})}{\pi(\boldsymbol{m}_s)\;q_{\text{LR}}(\widetilde{\boldsymbol{m}}_{s+1}|\,\boldsymbol{m}_s)}\right\}. \tag{2.54}$$

5. Generate a random number u_s from the uniform distribution $U[0,1]$.

6. If $u_s < \alpha(\boldsymbol{m}_s,\,\widetilde{\boldsymbol{m}}_{s+1})$, the proposed candidate is accepted as the new state in the chain, i.e., set

$$\boldsymbol{m}_{s+1} = \widetilde{\boldsymbol{m}}_{s+1}\,.$$

   If $u_s \geq \alpha(\boldsymbol{m}_s,\,\widetilde{\boldsymbol{m}}_{s+1})$, the proposed candidate is rejected and the current state is repeated in the chain, i.e., set

$$\boldsymbol{m}_{s+1} = \boldsymbol{m}_s\,.$$

7. Increase the chain counter by making $s = s+1$ and return to Step 3 until the desired chain length is achieved.

verge and asymptotically samples from the target distribution [35, 125]. The states which are generated before the convergence of the chain are regarded as the burn-in period. The states belonging to the burn-in period do not represent a sample of the target distribution, hence one must discard them. Therefore, when running a Markov chain, one must be able to determine the duration of the burn-in period. Usually, this is accomplished by monitoring the convergence of the Markov chain.

To monitor the performance of their Metropolis-Hastings MCMC sampling algorithm, Li and Reynolds [69, 68] resort to the common recommendation that one run several Markov chains in parallel. Furthermore, Li and Reynolds [69, 68] suggest monitoring the convergence of the chains by using the multivariate potential scale reduction factor (MPSRF) [14]. Each parallel chain is started from a distinct randomly selected initial state.

For a total of N_p parallel Markov chains, the MPSRF proceeds by first computing the within-sequence covariance matrix W_{N_s}, which is defined at the N_sth iteration, i.e., after N_sth state is generated in each chain, as [14]

$$W_{N_s} = \frac{1}{N_p(N_s-1)} \sum_{p=1}^{N_p} \sum_{s=1}^{N_s} \left(\boldsymbol{m}_{p,s} - \overline{\boldsymbol{m}}_{p,N_s}\right)\left(\boldsymbol{m}_{p,s} - \overline{\boldsymbol{m}}_{p,N_s}\right)^T . \tag{2.55}$$

In Eq. 2.55, $\boldsymbol{m}_{p,s}$, for $p = 1, 2, \ldots, N_p$ and $s = 1, 2, \ldots, N_s$, represents the sth sampled state of the pth parallel chain, and $\overline{\boldsymbol{m}}_{p,N_s}$ represents the mean of all N_s sampled states of the pth parallel chain, i.e.,

$$\overline{\boldsymbol{m}}_{p,N_s} = \frac{1}{N_s} \sum_{s=1}^{N_s} \boldsymbol{m}_{p,s} \qquad \text{for} \quad p = 1, 2, \ldots, N_p . \tag{2.56}$$

Next, one computes the between-sequence covariance matrix B_{N_s}, which is defined for the N_sth iteration as [14]

$$B_{N_s} = \frac{1}{N_p - 1} \sum_{p=1}^{N_p} \left(\overline{\boldsymbol{m}}_{p,N_s} - \overline{\boldsymbol{m}}_{N_s}\right)\left(\overline{\boldsymbol{m}}_{p,N_s} - \overline{\boldsymbol{m}}_{N_s}\right)^T , \tag{2.57}$$

where

$$\overline{\boldsymbol{m}}_{N_s} = \frac{1}{N_p} \sum_{p=1}^{N_p} \overline{\boldsymbol{m}}_{p,N_s} \ . \tag{2.58}$$

Subsequently, one computes the posterior covariance matrix V_{N_s} at the N_sth iteration as [14]

$$V_{N_s} = \frac{N_s - 1}{N_s} W_{N_s} + \frac{N_p + 1}{N_p} B_{N_s} \ . \tag{2.59}$$

Then, the MPSRF at the N_sth iteration, $\widehat{R}_{N_s}$, is computed as [14]

$$\widehat{R}_{N_s} = \frac{N_s - 1}{N_s} + \frac{N_p + 1}{N_p} \lambda_{N_s} \ , \tag{2.60}$$

where λ_{N_s} denotes the largest eigenvalue of the matrix $W_{N_s}^{-1} B_{N_s}$.

The factor $\widehat{R}_{N_s}$ approaches unity from above. When $\widehat{R}_{N_s}$ stabilizes close to unity, one considers that the chains have converged [14]. The underlying idea is that for $\widehat{R}_{N_s} \approx 1$, the within-sequence covariance matrix, W_{N_s}, and the posterior covariance matrix, V_{N_s}, are close [14], which implies that the parallel chains represent distinct samples of the same target distribution. Using the MPSRF method, the burn-in period is regarded as the period before the factor $\widehat{R}_{N_s}$ stabilizes. The states belonging to the burn-in period are discarded and the remaining states represent samples from the posterior pdf.

Covariance Matrix Adaptation:

As discussed earlier, the performance of many Metropolis-Hastings MCMC algorithms is greatly dependent on the choice of the proposal distribution. For many practical applications, it is difficult to correctly tune the selected proposal distribution to deliver an efficient MCMC algorithm. In opposition to a fixed proposal distribution, several previous works have proposed to instead automatically tune the chosen proposal distribution concomitantly with the MCMC sampling procedure.

Haario et al. [50, 51] suggested adapting the mean and the covariance matrix of a Gaussian proposal distribution based on the empirical mean and the empirical covariance

matrix which can be obtained using the states already proposed in the chain. Haario et al. [51] introduced a computationally efficient recursive formula to adapt the covariance matrix. Atchadé and Rosenthal [9] and Andrieu and Moulines [8] generalized the ideas of Haario et al. [51]. The works of Haario et al. [51], Atchadé and Rosenthal [9], Andrieu and Moulines [8], and Roberts and Rosenthal [103] discuss and prove convergence conditions for these adaptive approaches, see also Rosenthal [106] and Liang et al. [70].

The mean and covariance matrix adaptation proposed by Haario et al. [51], Atchadé and Rosenthal [9] and Andrieu and Moulines [8] are summarized in Liang et al. [70] as

$$\bar{\boldsymbol{\theta}}_{s+1} = \bar{\boldsymbol{\theta}}_s + \gamma_{s+1}(\boldsymbol{\theta}_{s+1} - \bar{\boldsymbol{\theta}}_s) \,, \tag{2.61}$$

and

$$C_{\Theta,s+1} = (1 - \gamma_{s+1})\, C_{\Theta,s} + \gamma_{s+1}\, (\boldsymbol{\theta}_{s+1} - \bar{\boldsymbol{\theta}}_s)\, (\boldsymbol{\theta}_{s+1} - \bar{\boldsymbol{\theta}}_s)^T \,, \tag{2.62}$$

In Eqs. 2.61 and 2.62, $\bar{\boldsymbol{\theta}}$ and C_Θ represent, respectively, the mean and the covariance matrix of the random vector $\boldsymbol{\theta}$, and the subscript s refers to the sth state in the chain. Also, the parameter γ_s is referred to as the leaning rate or the gain factor and should be selected to satisfy [70]

$$\sum_{s=1}^{\infty} \gamma_s = \infty \,, \tag{2.63}$$

and

$$\sum_{s=1}^{\infty} \gamma_s^{1+\delta} < \infty \,, \tag{2.64}$$

for some $\delta \in (0, 1]$. Haario et al. [51] suggested choosing $\gamma_s = 1/s$. The condition adopted by Haario et al. [51] can be generalized to $\gamma_s = O(1/s)$, and is referred to as diminishing adaptation.

Li and Reynolds [69, 68] suggested to adapt the GMM proposal distribution to further enhance the performance of their sampling algorithm. However, as pointed out by Rafiee and Reynolds [95], the mean of all the Gaussian distributions in the GMM proposal represent modes of the posterior pdf. Consequently, there is no need to adapt the mean of

the Gaussian distributions in the GMM proposal, only the covariance matrices need to be adapted. Since the GMM proposal distribution is independent of the current state in the chain, the diminishing adaptation can be relaxed [58]. Hence, following Rafiee and Reynolds [95], at the sth iteration in the chain, if $\boldsymbol{m}_{s+1}$ comes from the cth Gaussian in the GMM, we adapt only the covariance matrix of the cth Gaussian distribution in the GMM proposal using the following recursive rank-one covariance matrix adaptation:

$$C^*_{M,c,s+1} = (1-\gamma_{s+1})\, C^*_{M,c,s} + \gamma_{s+1}\, (\boldsymbol{m}_{s+1} - \boldsymbol{m}^*_c)\, (\boldsymbol{m}_{s+1} - \boldsymbol{m}^*_c)^T . \tag{2.65}$$

In Eq. 2.65, we adopt a constant learning rate of $\gamma_s = \mu_\gamma / N_m^2$, as suggested by Hansen [54, 53]. Furthermore, we tune the parameter μ_γ in such a way that the resulting MCMC algorithm delivers an acceptance rate between 0.20 and 0.50, as suggested by Roberts and Tweedie [104] and Roberts and Rosenthal [101]. Following Li and Reynolds [69, 68], we run several short length chains in order to select the parameter μ_γ which delivers the sought acceptance rate.

We emphasis that, each iteration of the Markov chain, we only adapt the covariance matrix of one Gaussian among the N_c Gaussian in the GMM approximation. The covariance matrix which is indeed adapted is the covariance matrix $C^*_{M,c,s}$ for the Gaussian from which the model $\boldsymbol{m}_{s+1}$ was actually sampled. Hence, in Eq. 2.65 it is understood that the model $\boldsymbol{m}_{s+1}$ is sampled from the cth Gaussian $\mathcal{N}(\boldsymbol{m}^*_c, C^*_{M,c})$ in the GMM proposal distribution.

2.4.2 Two-Level Markov Chain Monte Carlo Metropolis-Hastings Algorithm using the Distributed Gauss-Newton Method

The approach of Li and Reynolds [69, 68] requires a reservoir simulator with adjoint capability, which limits its application since most commercial reservoir simulators have limited or no adjoint solution capability. To circumvent this limitation, Rafiee and Reynolds [95, 93] extended the Li and Reynolds [69, 68] approach by using a modification of the DGN method [41] to compute the sensitivity matrix; their method does not require an adjoint solution.

The DGN modification introduced by Rafiee and Reynolds [95, 93] follows the original DGN method of Gao et al. [41]. However, to compute an approximation of the sensitivity matrix for a particular reservoir model $\boldsymbol{m}_p$, Rafiee and Reynolds [95, 93] suggested selecting the $N_c < N_m$ models in the data set, $\boldsymbol{m}_c$, for $c = 1, 2, \ldots, N_c$, which are the closest to the model $\boldsymbol{m}_p$, and computing an approximate sensitivity matrix by solving the underdetermined problem

$$(\Delta M^p)^T \left(G_{h,A}(\boldsymbol{m}_p)\right)^T = \left(\Delta D^p_{d_h}\right)^T . \tag{2.66}$$

Similarly to Eq. 2.33, in Eq. 2.66, $\Delta D^p_{d_h}$ denotes the $N_{d_h} \times N_c$ matrix whose cth column is given by $\boldsymbol{d}_h(\boldsymbol{m}_c) - \boldsymbol{d}_h(\boldsymbol{m}_p)$, for $c = 1, 2, \ldots, N_c$, and ΔM^p denotes the $N_m \times N_c$ matrix whose cth column is given by $\boldsymbol{m}_c - \boldsymbol{m}_p$, for $c = 1, 2, \ldots, N_c$. As before, note the entry in ith row, for $i = 1, 2, \cdots, N_{d_h}$, and cth column, for $c = 1, 2, \cdots, N_c$, of matrix $\Delta D^p_{d_h}$ is given by $d_{h,i}(\boldsymbol{m}_c) - d_{h,i}(\boldsymbol{m}_p)$, where $d_{h,i}$ denotes the ith entry component, for $i = 1, 2, \cdots, N_{d_h}$, of the column vector of predicted data, $\boldsymbol{d}_h$. Similarly, the entry in jth row, for $j = 1, 2, \cdots, N_m$, and cth column, for $c = 1, 2, \cdots, N_c$, of matrix ΔM^p is given by $m_{c,j} - m_{p,j}$, where $m_{c,j}$ is the jth entry of the column vector $\boldsymbol{m}_c$, for $c = 1, 2, \cdots, N_c$, and $m_{p,j}$ is the jth entry of the column vector $\boldsymbol{m}_p$.

To solve Eq. 2.66, Rafiee and Reynolds [95, 93] applied the singular value decomposition (SVD) [46] to the transpose of matrix ΔM^p, which results in

$$(\Delta M^p)^T = U \Lambda V^T . \tag{2.67}$$

In Eq. 2.67, the $N_m \times N_m$ matrix U contains in its columns the left singular vectors of $(\Delta M^p)^T$, the $N_c \times N_c$ matrix V contains in its columns the right singular vectors of $(\Delta M^p)^T$, and the $N_m \times N_c$ diagonal matrix Λ contains the singular values of $(\Delta M^p)^T$ on its diagonal. Representing the singular values as λ_c, for $c = 1, 2, \ldots, N_c$, the diagonal elements of the matrix Λ are ordered such that $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_{N_c}$. To compute the pseudo-inverse of $(\Delta M^p)^T$, Rafiee and Reynolds [95, 93] resorted to the truncated singular value decomposition

(TSVD) [55]

$$(\Delta M^p)^T \cong U_p \, \Lambda_p \, V_p^T \, , \tag{2.68}$$

where the $N_m \times N_p$ matrix U_p is formed using the first p columns of the matrix U, the $N_c \times N_p$ matrix V_p is formed using the first p columns of the matrix V, and the $N_p \times N_p$ diagonal matrix Λ_p has as its pth diagonal entry the pth highest singular values of the matrix $(\Delta M^p)^T$. Rafiee and Reynolds [95, 93] suggested that we chose p such that the relation λ_p/λ_1 is less than an user given threshold. From Eq. 2.68, the pseudo-inverse of $(\Delta M^p)^T$ is given by

$$(\Delta M^p)^{T+} \cong U_p \, \Lambda_p^{-1} \, V_p^T \, . \tag{2.69}$$

Using Eq. 2.69, the approximate sensitivity matrix is computed from Eq. 2.66 as

$$(G_{h,A}(\boldsymbol{m}_p))^T = U_p \, \Lambda_p^{-1} \, V_p^T \, \left(\Delta D_{d_h}^p\right)^T \, . \tag{2.70}$$

Using Eq. 2.70 to compute the approximate sensitivity matrix for any given model $\boldsymbol{m}_p$, Rafiee and Reynolds [95, 93] followed the same procedure for the DGN method described earlier in order to find modes of the posterior pdf. Similar to Li and Reynolds [69, 68], to quantify the uncertainty in reservoir predictions, Rafiee and Reynolds [95, 93] constructed a GMM approximation of the posterior pdf with the modes found, then used this approximation as the proposal distribution in a Metropolis-Hastings MCMC sampling framework.

CHAPTER 3

COMBINING LEAST-SQUARES SUPPORT VECTOR REGRESSION AND MARKOV CHAIN MONTE CARLO FOR HISTORY MATCHING AND UNCERTAINTY QUANTIFICATION

In this chapter, we describe the proposed methodology for combining the LS-SVR method and the Metropolis-Hastings MCMC sampling algorithm for history matching and uncertainty quantification. We employ a similar approach of the two-level MCMC algorithm introduced by Li and Reynolds [69, 68]. However, we replace the reservoir simulator by a LS-SVR proxy when evaluating the Metropolis-Hastings acceptance probability and this makes the method orders of magnitude faster than the method of both Li and Reynolds [69, 68] and Rafiee and Reynolds [95, 93]. Moreover, we first use the analytical gradient of the LS-SVR proxy model to find modes of the posterior probability density function (pdf), and then, the modes are used to construct an approximation of the posterior pdf. Finally, we generate a Markov chain using the GMM approximation of the posterior pdf as the proposal distribution. During the MCMC sampling, we replace the required reservoir simulation runs by the LS-SVR proxy model predictions. The details are presented throughout this chapter.

3.1 Least-Squares Support Vector Regression Formulation

As discussed earlier in this dissertation, the support vector regression method employs a suite of input vectors and its correspondent outputs, which are referred to as a training set, to construct an approximation of some unknown function, which can be used to compute outputs for input vectors not present in the original training set. In this research, we adopt the LS-SVR as introduced by Suykens and Vandewalle [119] and Suykens [117].

For application in the oil industry, the unknown function represents the reservoir

simulator. In this case, the input is given by the vector of reservoir simulator parameters, and the output represents the reservoir simulator predictions. Normally, in a conventional reservoir simulator one can easily format and establish the given inputs and the corresponding needed predictions. However, the LS-SVR proxy model is constructed based on a training set, which means that the composition of both input vectors and corresponding outputs is fixed beforehand. Thus, the resulting proxy provides predictions related to the kind of output used in the training procedure. Whenever one needs to predict a different type of output, unfortunately a new proxy model needs to be trained, which may require additional reservoir simulation runs to compute the necessary outputs, if these are not available. As will be clear in the following explanations, we intend to use the proxy predictions for both the history matching, where we estimate modes of the posterior pdf, and forecasting periods, where we evaluate the Metropolis-Hastings acceptance probability. Consequently, even though during the history matching process we only need the predictions which correspond to the observed data, we need to train the LS-SVR proxy model for both the history matching and forecasting periods. Therefore, we use an augmented vector of reservoir predictions to train the proxy model, the augmented vector includes the forecasting period. Accordingly, we need to extend the time of the reservoir simulator runs, which is used to generate the training set of inputs and outputs, to include the forecast period. We denote the reservoir simulation predictions corresponding to the forecasting period as the $N_{d_f}-$dimensional column vector $\boldsymbol{d}_f$. Hence, the augmented vector of reservoir predictions, denoted as $\boldsymbol{d}$, which is used to train the LS-SVR proxy model is defined as

$$\boldsymbol{d} = [\ \boldsymbol{d}_h^T,\ \boldsymbol{d}_f^T\ ]^T = g(\boldsymbol{m})\ . \tag{3.1}$$

In Eq. 3.1, the function $g(\boldsymbol{m})$ represents the reservoir predictions, generated with the reservoir simulator model $\boldsymbol{m}$, for both the history matching and forecasting periods. In this dissertation, we refer to dimension of the vector $\boldsymbol{d}$ as $N_d = N_{d_h} + N_{d_f}$, where N_{d_h} represents the dimension of the vector $\boldsymbol{d}_h$, and N_{d_f} represents the dimension of the vector $\boldsymbol{d}_f$.

For a given relationship $\boldsymbol{d} = g(\boldsymbol{m})$ representing the reservoir simulator predictions, including both history-matching and forecasting periods, which relates the N_m-dimensional column vector of input parameters, $\boldsymbol{m}$, to the N_d-dimensional column vector of output data, $\boldsymbol{d}$, the LS-SVR regression method employs a given training set to generate an ensemble of N_d one-dimensional functions $\hat{g}_i(\boldsymbol{m})$, for $i = 1, 2, \ldots, N_d$. Each function $\hat{g}_i(\boldsymbol{m})$, for $i = 1, 2, \ldots, N_d$, represents one entry of the augmented vector of reservoir predictions, precisely, the corresponding ith entry. Concisely, we train an one-dimensional LS-SVR proxy model for each entry of the output vector $\boldsymbol{d}$. The entire ensemble of N_d one-dimensional LS-SVR proxy models is what we refer to, in this dissertation, as the LS-SVR proxy model.

We denote a given training set as $T_s = \{(\boldsymbol{m}_k, \boldsymbol{d}_k),\ k = 1, 2, \ldots, N_t\}$. The size of the training set is denoted as N_t, i.e., the training set contains N_t training examples in the form of a pair of input and corresponding output vectors, $(\boldsymbol{m}_k, \boldsymbol{d}_k)$, for $k = 1, 2, \ldots, N_t$. For each training example, it holds that $\boldsymbol{d}_k = g(\boldsymbol{m}_k)$, for $k = 1, 2, \ldots, N_t$, i.e., the vector $\boldsymbol{d}_k$ is obtained by running the reservoir simulator, $g(\cdot)$, for the corresponding input vector $\boldsymbol{m}_k$, for $k = 1, 2, \ldots, N_t$.

For most real applications, usually both the input vector, $\boldsymbol{m}$, and the output vector, $\boldsymbol{d}$, are composed of different physical quantities, which may have different numerical scales. Significantly different numerical orders of magnitude between the entries of the vectors may result in a biased proxy model, i.e., the training process may be dominated by the physical quantities which have larger numerical values. To circumvent this numerical issue, usually the entries in both input and output vectors are normalized before the training procedure. In this dissertation, we adopt the usual normalization between zero and one, as described next.

We defined $\boldsymbol{m}^{\text{low}}$ by

$$\boldsymbol{m}^{\text{low}} \ = \ [m_j^{\text{low}}]_{j=1}^{N_m} \ \equiv \ [m_1^{\text{low}}, m_2^{\text{low}}, \ldots, m_{N_m}^{\text{low}}]^T \, ,$$

which is the N_m-dimensional column vector formed by selecting the corresponding lowest

entries in each component of the set of all input vectors that compose a given training set. Hence, the jth entry of the vector $\boldsymbol{m}^{\text{low}}$, for $j = 1, 2, \ldots, N_m$, represents the lowest value among all jth entries from all the vectors $\boldsymbol{m}_k$, for $k = 1, 2, \ldots, N_t$, i.e., for $j = 1, 2, \ldots, N_m$, the jth entry of the vector $\boldsymbol{m}^{\text{low}}$, denoted by m_j^{low}, is given by

$$m_j^{\text{low}} = \underset{1 \leq k \leq N_t}{\arg\min} \left\{ m_{k,j} \right\},$$

where $m_{k,j}$ represents the jth entry of the vector $\boldsymbol{m}_k$, for $k = 1, 2, \ldots, N_t$.

Similarly, we defined $\boldsymbol{m}^{\text{up}}$ by

$$\boldsymbol{m}^{\text{up}} \; = \; [m_j^{\text{up}}]_{j=1}^{N_m} \; \equiv \; [m_1^{\text{up}}, \, m_2^{\text{up}}, \, \ldots, \, m_{N_m}^{\text{up}}]^T \, ,$$

which is the N_m-dimensional column vector formed by selecting the corresponding largest entries in each component of the set of all input vectors that compose a given training set. Hence, the jth entry of the vector $\boldsymbol{m}^{\text{up}}$, for $j = 1, 2, \ldots, N_m$, represents the largest value among all jth entries from all the vectors $\boldsymbol{m}_k$, for $k = 1, 2, \ldots, N_t$, i.e., for $j = 1, 2, \ldots, N_m$, the jth entry of the vector $\boldsymbol{m}^{\text{up}}$, denoted by m_j^{up}, is given by

$$m_j^{\text{up}} = \underset{1 \leq k \leq N_t}{\arg\max} \left\{ m_{k,j} \right\},$$

where again $m_{k,j}$ represents the jth entry of the vector $\boldsymbol{m}_k$, for $k = 1, 2, \ldots, N_t$.

For the output vectors, we proceed in a similar manner. Specifically, we define

$$\boldsymbol{d}^{\text{low}} \; = \; [d_i^{\text{low}}]_{i=1}^{N_d} \; \equiv \; [d_1^{\text{low}}, \, d_2^{\text{low}}, \, \ldots, \, d_{N_d}^{\text{low}}]^T$$

as the N_d-dimensional column vector formed by selecting the corresponding lowest entries in all output vectors that compose a given training set, i.e., the ith entry of the vector $\boldsymbol{d}^{\text{low}}$, for $i = 1, 2, \ldots, N_d$, represents the lowest value among all ith entries from all the vectors

$\boldsymbol{d}_k$, for $k = 1, 2, \ldots, N_t$, in a given training set. Finally, we define

$$\boldsymbol{d}^{\text{up}} \;=\; [d_i^{\text{up}}]_{i=1}^{N_d} \;\equiv\; [d_1^{\text{up}}, d_2^{\text{up}}, \ldots, d_{N_d}^{\text{up}}]^T$$

as the N_d-dimensional column vector formed by selecting the corresponding largest entries in all output vectors that compose a given training set, i.e., the ith entry of the vector $\boldsymbol{d}^{\text{low}}$, for $i = 1, 2, \ldots, N_d$, represents the largest value among all ith entries from all the vectors $\boldsymbol{d}_k$, for $k = 1, 2, \ldots, N_t$, in a given training set.

We denote by $\tilde{\boldsymbol{m}}_k$, for $k = 1, 2, \ldots, N_t$, the kth normalized input vector of reservoir parameters, and as $\tilde{\boldsymbol{d}}_k$, for $k = 1, 2, \ldots, N_t$, the kth normalized vector of reservoir predictions. The entries of the normalized vectors $\tilde{\boldsymbol{m}}_k$ and $\tilde{\boldsymbol{d}}_k$ are computed, respectively, by

$$\tilde{m}_{k,j} = \frac{m_{k,j} \;-\; m_j^{\text{low}}}{m_j^{\text{up}} \;-\; m_j^{\text{low}}}\,, \quad \text{for} \quad j = 1, 2, \ldots, N_m \quad \text{and} \quad k = 1, 2, \ldots, N_t\,, \tag{3.2}$$

and

$$\tilde{d}_{k,i} = \frac{d_{k,i} \;-\; d_i^{\text{low}}}{d_i^{\text{up}} \;-\; d_i^{\text{low}}}\,, \qquad \text{for} \quad i = 1, 2, \ldots, N_d \quad \text{and} \quad k = 1, 2, \ldots, N_t\,. \tag{3.3}$$

In Eqs. 3.2 and 3.3, $\tilde{m}_{k,j}$ and $m_{k,j}$ represent, respectively, the jth entry of the vectors $\tilde{\boldsymbol{m}}_k$ and $\boldsymbol{m}_k$, similarly, $\tilde{d}_{k,i}$ and $d_{k,i}$ represent, respectively, the ith entry of the vectors $\tilde{\boldsymbol{d}}_k$ and $\boldsymbol{d}_k$. Clearly, the entries of the vectors $\tilde{\boldsymbol{m}}_k$ and $\tilde{\boldsymbol{d}}_k$ are between zero and one.

We represent the normalized training set as $\tilde{T}_s = \{(\tilde{\boldsymbol{m}}_k, \tilde{\boldsymbol{d}}_k),\ k = 1, 2, \ldots, N_t\}$. For the normalized training set, it holds that

$$\tilde{\boldsymbol{d}}_k = \tilde{g}(\tilde{\boldsymbol{m}}_k)\,. \tag{3.4}$$

In Eq. 3.4, $\tilde{g}(\cdot)$ represents the unknown relationship between the normalized vectors $\tilde{\boldsymbol{d}}_k$ and $\tilde{\boldsymbol{m}}_k$.

With the normalization, we now construct a LS-SVR proxy model for the function

$\tilde{g}(\cdot)$, using the normalized training set. The overall training procedure is presented in Appendix A. To apply the equations and procedure described in Appendix A for the normalized training set, one should set $\boldsymbol{y} \equiv \tilde{\boldsymbol{d}}$, $\boldsymbol{x} \equiv \tilde{\boldsymbol{m}}$ and $f(\boldsymbol{x}) \equiv \tilde{g}(\tilde{\boldsymbol{m}})$ in the equations of Appendix A, so that Eq. 3.4 coincides with Eq. A.1.

Following the procedure presented in Appendix A, the resulting LS-SVR model is given by (see Eq. A.16)

$$\hat{\tilde{g}}_i(\tilde{\boldsymbol{m}}) = \sum_{k=1}^{N_t} \alpha_{i,k}\, K_i(\tilde{\boldsymbol{m}}_k, \tilde{\boldsymbol{m}}) + b_i\,, \qquad \text{for} \quad i = 1,\, 2,\, \ldots\,,\, N_d\,. \tag{3.5}$$

In Eq. 3.5, the vector $\boldsymbol{\alpha}_i = [\alpha_{i,k}]_{k=1}^{N_t} = [\alpha_{i,1},\, \alpha_{i,2},\, \ldots\,,\, \alpha_{i,N_t}]^T$ is computed using Eq. A.13, b_i is computed using Eq. A.12, and the set of N_d functions $\hat{\tilde{g}}_i(\tilde{\boldsymbol{m}})$ represents the LS-SVR proxy approximation for the unknown normalized relationship $\tilde{\boldsymbol{d}} = \tilde{g}(\tilde{\boldsymbol{m}})$.

As discussed in Appendix A, $K_i(\tilde{\boldsymbol{m}}_k, \tilde{\boldsymbol{m}})$ in Eq. 3.5, for $i = 1,\, 2,\, \ldots\,,\, N_d$, represents a given kernel function. The kernel function has to be specified by the user [112]. Currently, there is no clear methodology available to identify the best choice of a kernel function for a given problem. When no further information is available, it is commonly recommended [112] that one use the radial basis function (RBF) kernel [16]. For any two given vectors $\tilde{\boldsymbol{m}}_k$ and $\tilde{\boldsymbol{m}}_\ell$, the RBF kernel is defined as

$$K_i(\tilde{\boldsymbol{m}}_k, \tilde{\boldsymbol{m}}_\ell) = \exp\left(-\frac{\|\tilde{\boldsymbol{m}}_k - \tilde{\boldsymbol{m}}_\ell\|_2^2}{\sigma_i^2}\right)\,. \tag{3.6}$$

In Eq. 3.6, $\|\cdot\|_2$ represents the Euclidean norm of a given vector, and σ_i represents a positive parameter defined beforehand by the user.

The RBF kernel sole parameter σ_i governs the complexity of the resulting function $\hat{\tilde{g}}_i(\tilde{\boldsymbol{m}})$. It is well established that a small values of σ_i leads to a more complex regression function. If a value of σ_i that is too small is used, the regression function can over-fit the data, which should be avoided, particularly for noisy data. Conversely, increasing the value of σ_i increases the smoothness of the corresponding support vector model. Typically, σ_i is

assumed to be correlated with the bounds of the input vector domain, i.e., it has the form

$$\sigma_i = c_{\sigma_i} \|\tilde{\boldsymbol{m}}^{\max} - \tilde{\boldsymbol{m}}^{\min}\|_2 \,, \qquad \text{for} \quad i = 1, 2, \ldots, N_d \,. \tag{3.7}$$

In Eq. 3.7, c_{σ_i} denotes a real-valued constant. Also in Eq. 3.7, $\tilde{\boldsymbol{m}}^{\max}$ represents the vector formed by selecting the maximum values among all normalized input vectors, i.e., the jth entry of $\tilde{\boldsymbol{m}}^{\max}$, for $j = 1, 2, \ldots, N_m$, is given by the maximum value among all the jth entries from all normalized input vectors $\tilde{\boldsymbol{m}}_k$, for $k = 1, 2, \ldots, N_t$. Similarly, $\tilde{\boldsymbol{m}}^{\min}$ represents the vector formed by selecting the minimum values among all normalized input vectors, i.e., the jth entry of $\tilde{\boldsymbol{m}}^{\min}$, for $j = 1, 2, \ldots, N_m$, is given by the minimum value among all the jth entries from all normalized input vectors $\tilde{\boldsymbol{m}}_k$, for $k = 1, 2, \ldots, N_t$. Since the vectors are normalized between zero and one, it holds that

$$\tilde{\boldsymbol{m}}^{\max} = [1, 1, \ldots, 1]^T \qquad \text{and} \qquad \tilde{\boldsymbol{m}}^{\min} = [0, 0, \ldots, 0]^T \,. \tag{3.8}$$

Consequently, its follows that Eq. 3.7 is equivalent to

$$\sigma_i = c_{\sigma_i} \sqrt{N_m} \,, \qquad \text{for} \quad i = 1, 2, \ldots, N_d \,. \tag{3.9}$$

A common recommendation is to use $c_{\sigma_i} = 0.50$, for $i = 1, 2, \ldots, N_d$. With this assumption, all the RBF kernel functions $K_i(\tilde{\boldsymbol{m}}_k, \tilde{\boldsymbol{m}}_\ell)$, for $i = 1, 2, \ldots, N_d$, which are used in Eq. 3.5, collapse to a single kernel function given by

$$K_i(\tilde{\boldsymbol{m}}_k, \tilde{\boldsymbol{m}}_\ell) \equiv K(\tilde{\boldsymbol{m}}_k, \tilde{\boldsymbol{m}}_\ell) = \exp\left(-\frac{\|\tilde{\boldsymbol{m}}_k - \tilde{\boldsymbol{m}}_\ell\|_2^2}{\sigma^2}\right) \,, \qquad \text{for} \quad i = 1, 2, \ldots, N_d \,, \tag{3.10}$$

where σ is computed using Eq. 3.9 with $c_\sigma = 0.50$, i.e., $\sigma = 0.50\sqrt{N_m}$.

From experience, we note that $c_\sigma = 0.50$ does not work well for every case. Actually, one could even adopt different c_{σ_i} for each one-dimensional LS-SVR proxy model $\hat{\tilde{g}}_i(\tilde{\boldsymbol{m}})$ in Eq. 3.5. However, different kernel functions require the computation and inversion of

different kernel matrices (see Appendix A), which increases the computational cost of the training procedure. In this research, we adopt a single kernel function for all one-dimensional LS-SVR proxy model. However, we use numerical experiments to determine an optimal value for c_σ, as we discuss later in this dissertation.

The main advantage a LS-SVR proxy model has is that the analytical expression of Eq. 3.5 can be evaluated in orders of magnitude less time than is required to run the reservoir simulator once. Furthermore, an analytical expression for the gradient of the LS-SVR proxy model can easily be computed from Eq. 3.5 as

$$\nabla_{\tilde{\boldsymbol{m}}}\, \hat{\tilde{g}}_i(\tilde{\boldsymbol{m}}) = \sum_{k=1}^{N_t} \frac{2\alpha_{i,k}}{\sigma^2}\, K(\tilde{\boldsymbol{m}}_k, \tilde{\boldsymbol{m}})(\tilde{\boldsymbol{m}}_k - \tilde{\boldsymbol{m}})\,, \qquad \text{for} \quad i = 1,\, 2,\, \ldots\,,\, N_d\,. \tag{3.11}$$

An analytical expression for computing the Hessian can also be found. The availability of an analytical gradient ceases the need for the adjoint solution, which allows a computational efficient application of a gradient-based optimization algorithm in the trust region minimization algorithm that we adopt in this research to find modes of the posterior pdf.

Obviously, the LS-SVR proxy model $\hat{\tilde{g}}_i(\tilde{\boldsymbol{m}})$, given by Eq. 3.5, which is constructed for the normalized relationship of Eq. 3.4, provides normalized values. To apply the LS-SVR proxy predictions, one need to convert the results using

$$\hat{g}_i(\boldsymbol{m}) = d_i^{\text{low}} + (d_i^{\text{up}} - d_i^{\text{low}}) \times \hat{\tilde{g}}_i(\tilde{\boldsymbol{m}})\,, \qquad \text{for} \quad i = 1,\, 2,\, \ldots\,,\, N_d\,. \tag{3.12}$$

In Eq. 3.12, $\hat{g}_i(\boldsymbol{m})$ denotes the LS-SVR proxy approximation of the corresponding reservoir prediction, $g_i(\boldsymbol{m})$, which includes both the historical and future prediction forecasts periods.

Similarly, Eq. 3.11 provides the gradient of $\hat{\tilde{g}}_i(\tilde{\boldsymbol{m}})$ with respect to $\tilde{\boldsymbol{m}}$, the required gradient of $\hat{g}_i(\boldsymbol{m})$ with respect to $\boldsymbol{m}$ is computed using

$$[\nabla_{\boldsymbol{m}}\, \hat{g}_i(\boldsymbol{m})]_j = \frac{[\nabla_{\tilde{\boldsymbol{m}}}\, \hat{\tilde{g}}_i(\tilde{\boldsymbol{m}})]_j \times (d_i^{\text{up}} - d_i^{\text{low}})}{m_j^{\text{up}} - m_j^{\text{low}}}\,, \qquad \text{for} \quad i = 1,\, 2,\, \ldots\,,\, N_d\,. \tag{3.13}$$

In Eq. 3.13, $[\nabla_{\boldsymbol{m}}\, \hat{g}_i(\boldsymbol{m})]_j$ and $[\nabla_{\tilde{\boldsymbol{m}}}\, \hat{\tilde{g}}_i(\tilde{\boldsymbol{m}})]_j$, for $j = 1,\, 2,\, \ldots\,,\, N_m$, represent, respectively, the

jth component of the gradients $\nabla_{\boldsymbol{m}}\,\hat{g}_i(\boldsymbol{m})$ and $\nabla_{\tilde{\boldsymbol{m}}}\,\hat{\tilde{g}}_i(\tilde{\boldsymbol{m}})$.

3.1.1 Influence of the Least-Squares-Support-Vector-Regression Training Parameters

As presented in the previous section and in Appendix A, the LS-SVR training procedure depends on two parameters, γ (see Eq. A.3) and σ (see Eq. 3.10). For the training procedure using normalized input and output vectors, a common recommendation on the literature is to adopt $\gamma = 200$ and use $c_\sigma = 0.50$ to compute σ (see Eq. 3.9).

The aim of this section is to investigate those recommendations and provide some insight on how the LS-SVR regression function works. For this purpose, we study the LS-SVR behavior using the one-dimensional function given by

$$\begin{aligned} f(x) = &- 0.00381\,x^6 + 0.1513\,x^5 - 2.223\,x^4 + 14.95\,x^3 - 45.0\,x^2 + 46.8\,x \\ &+ 0.50\,\sin(8.0\,x + 0.50\,\pi) + 10.0\,. \end{aligned} \tag{3.14}$$

To assess the effect of different values of γ and σ, we train a LS-SVR proxy model for the function $f(x)$, of Eq. 3.14, using the input x's presented in Table 3.1 to construct a training set. Then, the performance of the resulting LS-SVR is assessed using both the original training set and the test set examples presented in Table 3.2. To measure the performance of different values of γ and σ, we compute the root-mean-square error (RMSE) for the LS-SVR proxy predictions for both training and test examples presented in Tables 3.1 and 3.2. The RMSE is defined as

$$\text{RMSE} = \sqrt{\frac{\sum_{s=1}^{N_s}\left[\hat{f}(x_s) - f(x_s)\right]^2}{N_s}}\,. \tag{3.15}$$

In Eq. 3.15, N_s represent the total number of training examples ($N_s = 31$ for this case, see Table 3.1) or the total number of test examples ($N_s = 30$ for this case, see Table 3.2), x_s, for $s = 1, 2, \ldots, N_s$, denotes the sth training or test example, $\hat{f}(x_s)$ represents the proxy predictions for the sth example, and $f(x_s)$ the respective predictions of the function $f(x)$ of

Example	Input	Example	Input	Example	Input	Example	Input
1	0.000	9	2.667	17	5.333	25	8.000
2	0.333	10	3.000	18	5.667	26	8.333
3	0.667	11	3.333	19	6.000	27	8.667
4	1.000	12	3.667	20	6.333	28	9.000
5	1.333	13	4.000	21	6.667	29	9.333
6	1.667	14	4.333	22	7.000	30	9.667
7	2.000	15	4.667	23	7.333	31	10.000
8	2.333	16	5.000	24	7.667		

Table 3.1: Input x's used create a training set to construct the LS-SVR proxy for the function $f(x)$ of Eq. 3.14.

Test	Input	Test	Input	Test	Input	Test	Input	Test	Input
1	0.1667	7	2.1667	13	4.1667	19	6.1667	25	8.1667
2	0.5000	8	2.5000	14	4.5000	20	6.5000	26	8.5000
3	0.8333	9	2.8333	15	4.8333	21	6.8333	27	8.8333
4	1.1667	10	3.1667	16	5.1667	22	7.1667	28	9.1667
5	1.5000	11	3.5000	17	5.5000	23	7.5000	29	9.5000
6	1.8333	12	3.8333	18	5.8333	24	7.8333	30	9.8333

Table 3.2: Input x's used as test examples to assess the LS-SVR proxy for the function $f(x)$ of Eq. 3.14.

Eq. 3.14. No noise is added to the predictions of $f(x_s)$ values.

In Figs. 3.1 and 3.2 we present the LS-SVR proxy performance when varying parameter γ, and for six different values of parameter σ, $\sigma = 0.040$ (red curve), $\sigma = 0.062$ (blue curve), $\sigma = 0.080$ (gray curve), $\sigma = 0.120$ (orange curve), $\sigma = 0.160$ (magenta curve), and $\sigma = 0.200$ (green curve). In Fig. 3.1 we present the RMSE computed using the training set examples presented in Table 3.1, and in Fig. 3.2 we present the results using the test set examples presented in Table 3.2. As one can see from Figs. 3.1 and 3.2, the effect of increasing the value of γ is to reduce the value of the respective RMSE, i.e., by increasing γ one obtain a better match of the data, as one can see in Fig. 3.1, which presents the results for the training set examples. Furthermore, notice that the overall performance does not vary considerably for values of γ greater than $\gamma = 800$. Also, notice that the common recommendation of $\gamma = 200$ is somehow conservative. Although this conclusion is based on the data presented on Figs. 3.1 and 3.2, we observed similar behavior in all LS-SVR proxy trained in

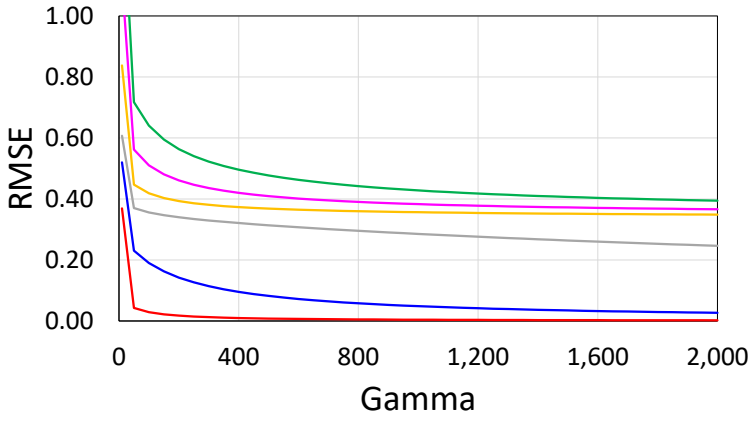


Figure 3.1: Effect of parameter γ on the LS-SVR proxy model performance for different values of parameter σ: $\sigma = 0.040$ (red curve), $\sigma = 0.062$ (blue curve), $\sigma = 0.080$ (gray curve), $\sigma = 0.120$ (orange curve), $\sigma = 0.160$ (magenta curve), and $\sigma = 0.200$ (green curve). RMSE computed using the training set data presented in Table 3.1.

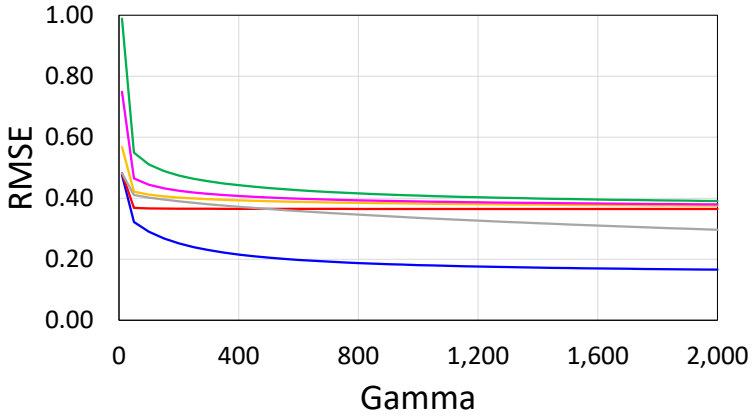
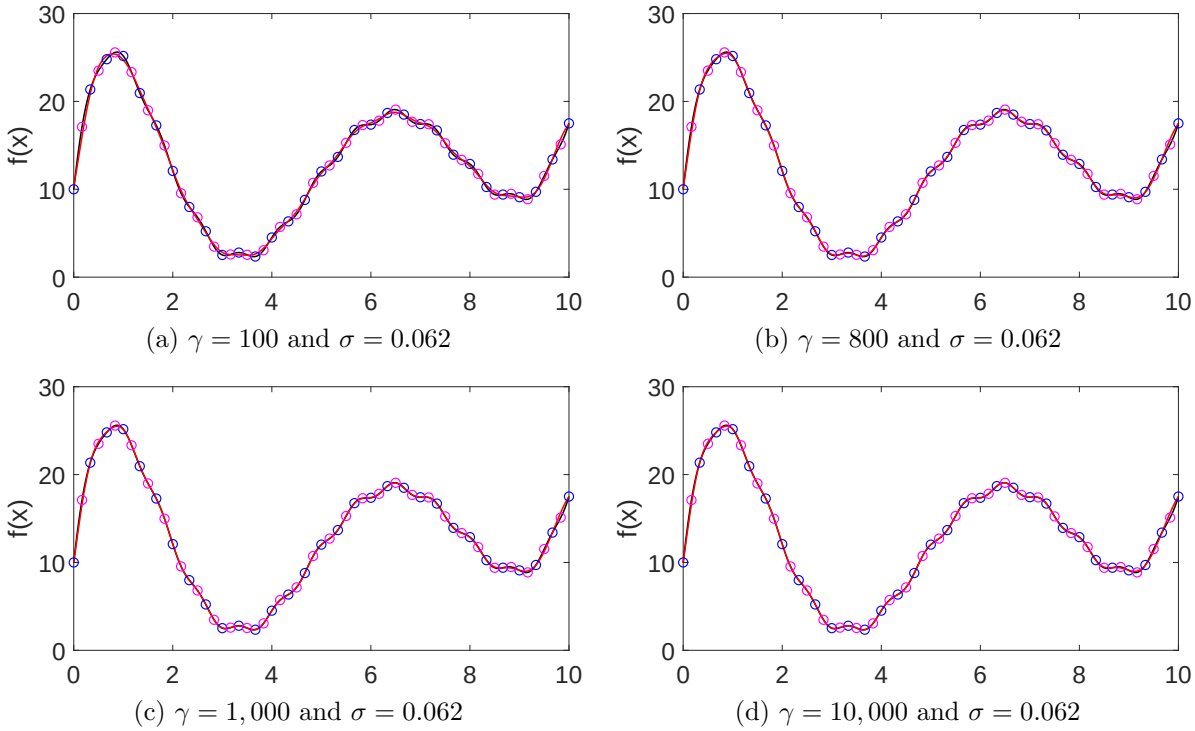


Figure 3.2: Effect of parameter γ on the LS-SVR proxy model performance for different values of parameter σ: $\sigma = 0.040$ (red curve), $\sigma = 0.062$ (blue curve), $\sigma = 0.080$ (gray curve), $\sigma = 0.120$ (orange curve), $\sigma = 0.160$ (magenta curve), and $\sigma = 0.200$ (green curve). RMSE computed using the test set data presented in Table 3.2.



(a) $\gamma = 100$ and $\sigma = 0.062$

(b) $\gamma = 800$ and $\sigma = 0.062$

(c) $\gamma = 1,000$ and $\sigma = 0.062$

(d) $\gamma = 10,000$ and $\sigma = 0.062$

Figure 3.3: True predictions of function $f(x)$ of Eq. 3.14 compared to the LS-SVR proxy predictions for $\sigma = 0.062$ and different values of γ ($\gamma = 100$, $\gamma = 800$, $\gamma = 1,000$, and $\gamma = 10,000$). In figures (a) through (d) are presented the true function $f(x)$ of Eq. 3.14 (black curve), the LS-SVR proxy predictions (red curve), the training set examples (blue circles), and the test set examples (magenta circles).

this dissertation. Some argue that increasing the value of γ could result in over-fitting the data. However, we observed that increasing the value of γ has the sole effect of improving the data match. This is the behavior observed in Fig. 3.3, in which we compare the function $f(x)$ of Eq. 3.14 with the LS-SVR proxy trained for $\sigma = 0.062$ and four different values of γ, $\gamma = 100$, $\gamma = 800$, $\gamma = 1,000$, and $\gamma = 10,000$. Notice that increasing the value of γ results in a better data match, and even for the extremely high value of $\gamma = 10,000$ in Fig. 3.3(d) no over-fitting behavior is observed. Based on the results presented, we recommend and adopt a fixed value of $\gamma = 800$ in all LS-SVR training procedure conducted in this dissertation.

In Figs. 3.4 and 3.5 we present the LS-SVR proxy performance when varying parameter σ, and for six different values of parameter γ, $\gamma = 50$ (blue curve), $\gamma = 200$ (red curve), $\gamma = 400$ (gray curve), $\gamma = 800$ (orange curve), $\gamma = 1,000$ (magenta curve), and $\gamma = 2,000$ (green curve). In Fig. 3.4 we present the RMSE computed using the training set examples

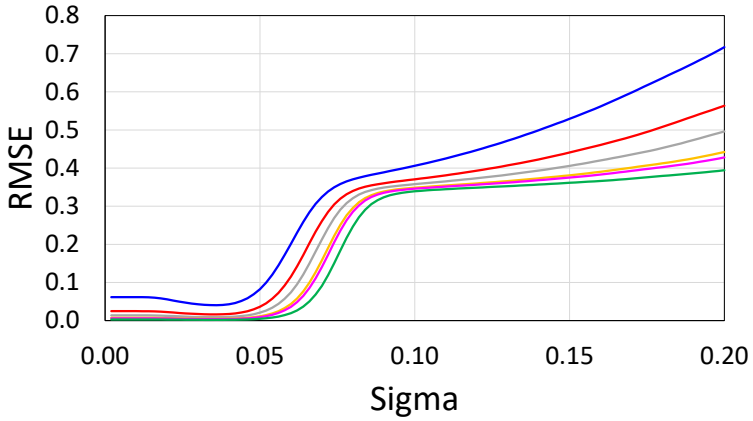


Figure 3.4: Effect of parameter σ on the LS-SVR proxy model performance for different values of parameter γ: $\gamma = 50$ (blue curve), $\gamma = 200$ (red curve), $\gamma = 400$ (gray curve), $\gamma = 800$ (orange curve), $\gamma = 1,000$ (magenta curve), and $\gamma = 2,000$ (green curve). RMSE computed using the training set data presented in Table 3.1.

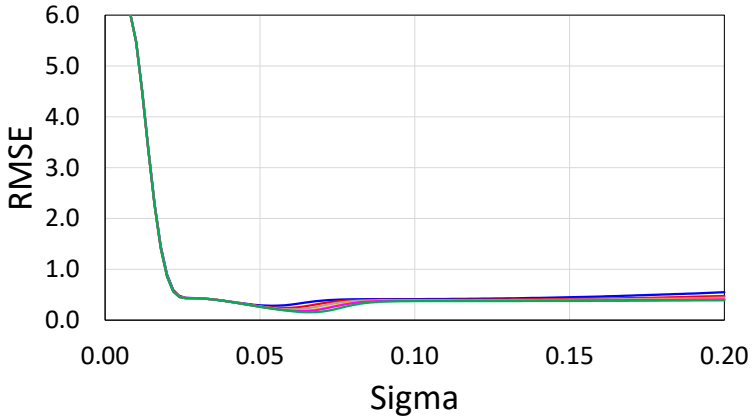
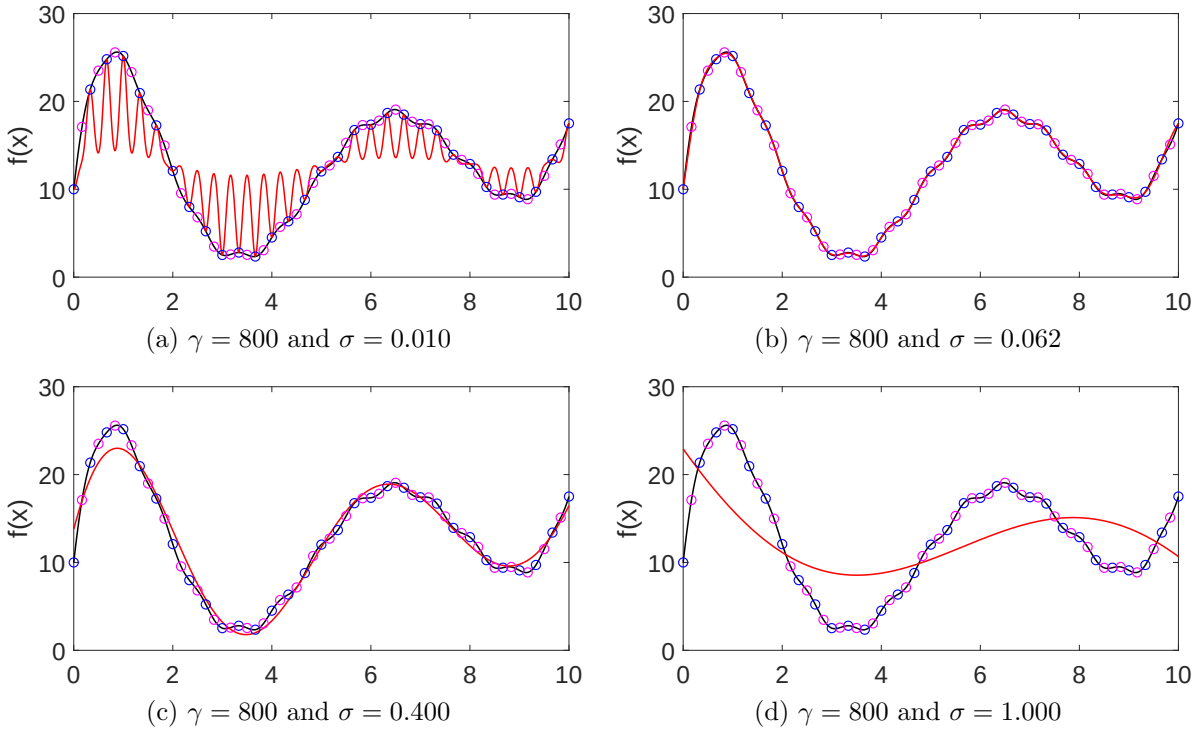


Figure 3.5: Effect of parameter σ on the LS-SVR proxy model performance for different values of parameter γ: $\gamma = 50$ (blue curve), $\gamma = 200$ (red curve), $\gamma = 400$ (gray curve), $\gamma = 800$ (orange curve), $\gamma = 1,000$ (magenta curve), and $\gamma = 2,000$ (green curve). RMSE computed using the test set data presented in Table 3.2.

presented on Table 3.1, and in Fig. 3.5 we present the results using the test set examples presented on Table 3.2. The behavior that the LS-SVR proxy presented in Fig. 3.4 reflects the fact that small values of σ leads to over-fitting the data, thus the value of the computed RMSE using the training set data will reduce, but when the resulting proxy is applied to the test set data, we get a large RMSE, as shown in Fig. 3.5, reflecting the fact that the over-fitting regression function cannot represent the test set data properly. Conversely, increasing the value of σ leads to a smooth LS-SVR proxy, which results in an increasing RMSE value. A more interesting behavior is observed on Fig. 3.5, in which the test set data is used, as one can observe, all RMSE curves have a minimum. A small value of RMSE in this case translates to a good LS-SVR proxy model, i.e., the LS-SVR proxy provides good predictions for data not present in the original training set. Therefore, one should select the optimal value of σ by finding the value which minimizes the RMSE (or some other measure) for some given test set. Nevertheless, notice that for $\gamma = 800$, the minimum value occurs at $\sigma = 0.062$, which for the normalizing training procedure corresponds to a $c_\sigma = 0.062$. Hence, for this particular case, the optimal value of c_σ is almost one order of magnitude different from the common recommendation $c_\sigma = 0.50$. Based on this results, we recommend that one select an optimal value of c_σ by minimizing the RMSE for a given test set. To accomplish this, we adopt numerical experiments, which consist of generating the RMSE data for several values of σ and selecting the minimum based on the generated data. Finally, in Fig. 3.6, we compare the function $f(x)$ of Eq. 3.14 with the LS-SVR proxy trained for $\gamma = 800$ and four different values of σ, $\sigma = 0.010$, $\sigma = 0.062$, $\sigma = 0.400$, and $\sigma = 1.000$, in which one can observe the over-fitting behavior for the case with $\sigma = 0.010$, see Fig. 3.6(a), the optimal fitting for $\sigma = 0.062$, see Fig. 3.6(b), and the smooth LS-SVR proxy as we increase the value of σ, such as $\sigma = 1.000$, see Fig. 3.6(d).

3.2 History Matching and Uncertainty Quantification using the Least-Squares Support Vector Regression Proxy and Markov Chain Monte Carlo

In this research, we follow Li and Reynolds [69, 68] and Rafiee and Reynolds [95, 93]



(a) $\gamma = 800$ and $\sigma = 0.010$

(b) $\gamma = 800$ and $\sigma = 0.062$

(c) $\gamma = 800$ and $\sigma = 0.400$

(d) $\gamma = 800$ and $\sigma = 1.000$

Figure 3.6: True predictions of function $f(x)$ of Eq. 3.14 compared to the LS-SVR proxy predictions for $\gamma = 800$ and different values of σ ($\sigma = 0.001$, $\sigma = 0.062$, $\sigma = 0.400$, and $\sigma = 1.000$). In figures (a) through (d) are presented the true function $f(x)$ of Eq. 3.14 (black curve), the LS-SVR proxy predictions (red curve), the training set examples (blue circles), and the test set examples (magenta circles).

and construct a GMM approximation of the posterior pdf to use as proposal distribution in a Metropolis-Hastings MCMC sampling framework. The GMM approximation is constructed centered at some modes of the posterior pdf for reservoir model parameters. However, our objective is to achieve computational efficiency when applying MCMC to quantify uncertainty in reservoir simulator predictions. To accomplish this goal, we adopt a LS-SVR proxy model as a suitable replacement to the reservoir simulator. The use of the LS-SVR proxy model is twofold. Firstly, similarly to Li and Reynolds [69, 68] and Rafiee and Reynolds [95, 93], we conduct several parallel gradient-based minimization problems in order to find modes of the posterior pdf $\pi(\boldsymbol{m})$, of Eq. 2.6, to construct a GMM approximation. However, we make use of the analytical gradient of the LS-SVR proxy model, as presented in Eqs. 3.11 and 3.13, to compute an approximation of the required sensitivity matrix $G(\boldsymbol{m})$ of Eq. 2.10. Consequently, our approach does not require a reservoir simulator which is able to solve the

adjoint problem. Second, different from Li and Reynolds [69, 68] and Rafiee and Reynolds [95, 93], we advocate the use of the LS-SVR proxy model as a replacement of the real reservoir simulator during the MCMC sampling procedure. Inasmuch as the LS-SVR proxy model is faster to evaluate when compared to the the real reservoir simulator, our method results in a significant reduction in the total computational time required to apply the MCMC sampling process, which leads to a computational efficient MCMC sampling algorithm.

One crucial aspect of the use of a proxy model is the accuracy of the adopted proxy predictions. The LS-SVR proxy adopted here is constructed based on a given training set. Consequently, the accuracy of the resulting LS-SVR proxy depends on the training examples which are selected to compose the training set. However, it is extremely difficult to determine before hand which training examples should be included in the training set to achieve the desired accuracy for any given application. To address this difficulty, we propose here to start with an initial training set, then modify and adapt it during the minimization process which is conducted to find the modes of the posterior pdf. The strategy adopted is to include in the original training set the updates obtained when performing the minimization problems. The underlying main idea is that as the iterates of the multiple minimization problems approach the modes, the iterates added to the training set will come from regions of increasing probability of the posterior pdf. Furthermore, the points added to the training set will be concentrate around the modes of the posterior pdf. As a consequence, the LS-SVR proxy which is trained with such a training set is expected to deliver accurate predictions around the modes of the posterior pdf. This particular property of the resulting LS-SVR proxy is precisely the desirable feature for a proxy intended to replace the reservoir simulator in a MCMC sampling framework. In the next sections, we delineate the details of how to construct and adapt the training set, as well as the details of our proposed Metropolis-Hastings MCMC sampling algorithm.

3.2.1 First Step: Finding Modes of the Posterior Probability Density Function

We intend to construct a GMM distribution centered at the modes of the posterior pdf

$\pi(\boldsymbol{m})$ of of Eq. 2.6. Therefore, we need to find modes of the posterior pdf, which means we need an affordable process to find several minima of the objective function $O(\boldsymbol{m})$ of Eq. 2.7. As discussed earlier, normally the vector of model parameters $\boldsymbol{m}$ is composed with different reservoir properties [98], and the number of parameters is very large. As a consequence, the direct minimization $O(\boldsymbol{m})$ of Eq. 2.7 can result in an ill-conditioned problem. To circumvent this issue, we follow Rafiee and Reynolds [95] and adopt the dimensionless transformation of the vectors $\boldsymbol{m}$ and $\boldsymbol{d}_h$ as presented, respectively, on Eqs. 2.11 and 2.12. Therefore, the minimization process to find modes of the posterior pdf is actually conducted by minimizing the dimensionless objective function $O(\hat{\boldsymbol{m}})$ as defined on Eq. 2.13. As discussed earlier, both $O(\boldsymbol{m})$, of Eq. 2.7, and $O(\hat{\boldsymbol{m}})$, of Eq. 2.13, share the same minima, when both $\boldsymbol{m}$ and $\hat{\boldsymbol{m}}$ are related by Eq. 2.11.

As we present next, the minimization algorithm used in this research to find minima of $O(\hat{\boldsymbol{m}})$, of Eq. 2.13, requires the the gradient, $\nabla_{\hat{\boldsymbol{m}}} O(\hat{\boldsymbol{m}})$, and Hessian, $\mathcal{H}(\hat{\boldsymbol{m}})$, of the dimensionless objective function. For our case, the gradient and Hessian are computed, respectively, by Eqs. 2.14 and 2.15. Furthermore, the required dimensionless sensitivity matrix, $G_D(\hat{\boldsymbol{m}})$, is computed using Eq. 2.17. Notice that to use Eq. 2.17, one must be able do assemble the sensitivity matrix $G(\boldsymbol{m})$, which can be accomplished by solving the adjoint problem, which was the procedure adopted by Li and Reynolds [69, 68]. Nevertheless, commercial reservoir simulators usually do not solve the adjoint problem, thus one needs to approximate the sensitivity matrix. As stated earlier, in this research we take advantage of the analytical gradient of the LS-SVR proxy model to approximate the sensitivity matrix $G(\boldsymbol{m})$ of Eq. 2.17, which then is given by (see Eq. 3.13)

$$G(\boldsymbol{m}) \cong G_{\text{app}}(\boldsymbol{m}) = \left[\nabla_{\boldsymbol{m}} \hat{g}_1(\boldsymbol{m}),\ \nabla_{\boldsymbol{m}} \hat{g}_2(\boldsymbol{m}),\ \ldots\ ,\ \nabla_{\boldsymbol{m}} \hat{g}_{N_{d_h}}(\boldsymbol{m})\right]^T, \tag{3.16}$$

where the $\hat{g}_i(\boldsymbol{m})$'s, for $i = 1, 2, \ldots, N_{d_h}$, and their gradients are evaluated based on the LS-SVR proxy.

In this research, the minimization algorithm used to find minima of $O(\hat{\boldsymbol{m}})$ of Eq. 2.13

is the trust-region minimization algorithm which is described in Appendix B. To apply the procedure and use the equations described in Appendix B, one should set $\boldsymbol{x} \equiv \hat{\boldsymbol{m}}$, $\boldsymbol{y}(\boldsymbol{x}) \equiv \hat{\boldsymbol{d}}_h(\hat{\boldsymbol{m}})$ and $O(\boldsymbol{x}) \equiv O(\hat{\boldsymbol{m}})$, so that Eqs. 2.13 and B.1 exactly correspond. Also, one should set $\mathcal{G}(\boldsymbol{x}) \equiv \mathcal{G}(\hat{\boldsymbol{m}})$, so that Eqs. 2.14 and B.2 correspond. Finally, one should set $\mathcal{H}(\boldsymbol{x}) \equiv \mathcal{H}(\hat{\boldsymbol{m}})$, so Eqs. 2.15 and B.3 become equivalent. The required sensitivity matrix is approximated using Eq. 3.16, as discussed above.

As discussed earlier, the aim in solving several minimization problems is to attempt to find modes of the posterior pdf. Let us assume that N_e modes are sought. In an effort to find the N_e modes, we randomly select N_e models by sampling the prior pdf. One concern here is how representative this initial sample of the prior pdf is. Although Li and Reynolds [69, 68] and Rafiee and Reynolds [95, 93] simply directly sampled the Gaussian prior pdf, we advocate taking advantage of some design of experiment technique would achieve a more representative initial sample. Therefore, we choose instead to select the initial sample from the prior pdf by using a well know statistical method called Latin hypercube sampling (LHS) (see McKay et al. [74]). The LHS sampling method tries to spread a predefined total number of samples to better cover the support of a given pdf, thus, one guarantees that even for a small sized sample, the tails of the pdf are represented, which for our applications better represent the original available information, i.e., the prior pdf.

After selecting the N_e initial samples from the prior pdf, we then conduct N_e minimization problems. Each minimization problem uses one of the N_e sampled models as its initial guess. Motivated by the work of Gao et al. [41] and Rafiee and Reynolds [95], we solve all N_e minimization problems simultaneously, iteration by iteration. Here, we regard the process of solving all N_e minimization problems as the minimization loop. At each iteration of the minimization loop, we perform exactly one iteration of the trust region algorithm (see Appendix B) for each minimization problem. By one iteration of the trust region we mean that we solve the trust-region sub-problem given by Eqs. B.5 and B.6 (see Appendix B). The reasons for solving the minimization problems simultaneously is to take advantage of the LS-SVR proxy model, as will become clear in the following explanation.

To compute the LS-SVR proxy gradient in Eq. 3.16, which is needed to solve the trust-region sub-problem, we need to define an initial training set to train an initial LS-SVR proxy model. One could use the already selected N_e initial guesses to train an initial LS-SVR proxy model. However, including additional models in the training set results in an initial LS-SVR proxy with improved predictions. Thus, in addition to the N_e selected initial guesses, we propose to sample additional N_a models from the prior pdf to compose the initial training set. As before, we adopt the LHS to sample these additional N_a models. In the applications presented in this dissertation, we found using a N_a value on the order of two to four times the total number of minimization problems, N_e, gives a good performance of the initial LS-SVR proxy. The resulting initial training set has $N_t = N_e + N_a$ training examples.

We start each iteration of the minimization loop by training a LS-SVR proxy model using the current available training set. The initial LS-SVR proxy is trained using the initial training set, as described above. Then, for all minimization problems in the loop, we solve the trust region sub-problem to compute the updated estimate of the minimum of $O(\hat{\boldsymbol{m}})$ of Eq. 2.13. The updated estimates are then added to the training set following a minimum distance procedure explained later in this dissertation. Notice that, we use the same LS-SVR proxy model at the same iteration of the minimization loop to approximate the required sensitivity matrix for all minimization problems in the loop. The updated training set from one iteration is then used to train a new LS-SVR proxy model for the next iteration of the minimization loop. After every iteration of the minimization loop, we add the updated estimates of the minima to the training set, until the convergence criteria are satisfied. Details of the minimization process are described next.

As described in Appendix B, at the nth minimization iteration and for each minimization problem in the minimization loop, we construct a quadratic approximation of the objective function $O(\hat{\boldsymbol{m}})$, of Eq. 2.13, around the current estimate of the minimum. To construct the quadratic approximation we use the LS-SVR proxy gradient to compute an approximation, $G_{\text{app}}(\boldsymbol{m})$, for the required sensitivity matrix, $G(\boldsymbol{m})$, as shown is Eq. 3.16.

Then we solve the trust-region minimization sub-problem given by

$$\underset{\delta\hat{\boldsymbol{m}}_e^{(n)}}{\text{minimize}} \;\; q_e^{(n)}(\delta\hat{\boldsymbol{m}}_e^{(n)}) = O(\hat{\boldsymbol{m}}_e^{(n)}) + \mathcal{G}(\hat{\boldsymbol{m}}_e^{(n)})^T \, \delta\hat{\boldsymbol{m}}_e^{(n)} \, + \, \frac{1}{2}\delta\hat{\boldsymbol{m}}_e^{(n)\,T} \, \mathcal{H}(\hat{\boldsymbol{m}}_e^{(n)}) \, \delta\hat{\boldsymbol{m}}_e^{(n)} \,, \quad (3.17)$$

for $e = 1, 2, \ldots, N_e$, subject to

$$\|\delta\hat{\boldsymbol{m}}_e^{(n)}\| \leq \delta_e^{(n)} \,. \quad (3.18)$$

In Eqs. 3.17 and 3.18, $\hat{\boldsymbol{m}}_e^{(n)}$ represents the current estimate of the minimum for the eth minimization problem at the nth iteration, $\delta\hat{\boldsymbol{m}}_e^{(n)}$ represents the search direction, $q_e^{(n)}(\delta\hat{\boldsymbol{m}}_e^{(n)})$ represents the quadratic approximation of $O(\hat{\boldsymbol{m}})$ for the eth minimization problem at the nth iteration, and $\delta_e^{(n)}$ represents the trust region radius for the eth minimization problem at the nth iteration (see Appendix B).

Also in Eq. 3.17, $O(\hat{\boldsymbol{m}}_e^{(n)})$, $\mathcal{G}(\hat{\boldsymbol{m}}_e^{(n)})$ and $\mathcal{H}(\hat{\boldsymbol{m}}_e^{(n)})$ represent, respectively, the objective function, its gradient and its Hessian, which are computed, respectively, by Eqs. 2.13, 2.14 and 2.15, for $\hat{\boldsymbol{m}} = \hat{\boldsymbol{m}}_e^{(n)}$ using the LS-SVR approximate sensitivity matrix $G_{\text{app}}(\boldsymbol{m})$ of Eq. 3.16. Hence, for each minimization problem, the required dimensionless sensitivity matrix is computed using Eq. 2.17 for the following approximate sensitivity matrix

$$G_{\text{app}}(\boldsymbol{m}_e^{(n)}) = \left[\nabla_{\boldsymbol{m}} \, \hat{g}_1(\boldsymbol{m}_e^{(n)}), \; \nabla_{\boldsymbol{m}} \, \hat{g}_2(\boldsymbol{m}_e^{(n)}), \; \ldots \, , \; \nabla_{\boldsymbol{m}} \, \hat{g}_{N_{d_h}}(\boldsymbol{m}_e^{(n)})\right]^T , \quad (3.19)$$

for $e = 1, 2, \ldots, N_e$. In Eq. 3.19, $\nabla_{\boldsymbol{m}} \, \hat{g}_i(\boldsymbol{m}_e^{(n)})$, for $i = 1, 2, \ldots, N_{d_h}$, is computed using Eq. 3.13 for $\boldsymbol{m} = \boldsymbol{m}_e^{(n)}$ in the current LS-SVR proxy model. Note that to use Eq. 2.17, one needs to compute $\boldsymbol{m}_e^{(n)}$, i.e., the corresponding dimensional counterpart of $\hat{\boldsymbol{m}}_e^{(n)}$ for the reservoir parameters vector, using

$$\boldsymbol{m}_e^{(n)} = \boldsymbol{m}_{\text{pr}} + C_M^{1/2} \, \hat{\boldsymbol{m}}_e^{(n)} \qquad \text{for} \quad e = 1, 2, \ldots, N_e \,. \quad (3.20)$$

We denote the solution of the eth trust region sub-problem given by Eqs. 3.17 and 3.18, at the nth iteration, as $\delta\hat{\boldsymbol{m}}_e^{(n)*}$. Then, we define the updated estimate of the minimum,

for the eth minimization problem at the nth iteration, as

$$\hat{\boldsymbol{m}}_{e,\text{upd}}^{(n)} = \hat{\boldsymbol{m}}_e^{(n)} + \delta\hat{\boldsymbol{m}}_e^{(n)*} \qquad \text{for} \quad e = 1, 2, \ldots, N_e \,. \tag{3.21}$$

The trust region radius, $\delta_e^{(n)}$, for the eth minimization problem at the nth iteration, is updated depending on the performance of the quadratic approximation $q_e^{(n)}(\delta\hat{\boldsymbol{m}}_e^{(n)})$, as described in Appendix B. The performance of $q_e^{(n)}(\delta\hat{\boldsymbol{m}}_e^{(n)})$ is measured by (see Eq. B.7)

$$\rho_e^{(n)}(\delta\hat{\boldsymbol{m}}_e^{(n)*}) = \frac{O(\hat{\boldsymbol{m}}_e^{(n)}) \;-\; O(\hat{\boldsymbol{m}}_{e,\text{upd}}^{(n)})}{q_e^{(n)}(\mathbf{0}) \;-\; q_e^{(n)}(\delta\hat{\boldsymbol{m}}_e^{(n)*})} \qquad \text{for} \quad e = 1, 2, \ldots, N_e \,. \tag{3.22}$$

Details on how to update the trust region radius are given in Appendix B (see Algorithm B.1).

Notice that to compute $O(\hat{\boldsymbol{m}}_{e,\text{upd}}^{(n)})$ in Eq. 3.22, we need to run the reservoir simulator for the corresponding reservoir model $\boldsymbol{m}_{e,\text{upd}}^{(n)}$, where $\hat{\boldsymbol{m}}_{e,\text{upd}}^{(n)}$ and $\boldsymbol{m}_{e,\text{upd}}^{(n)}$ are related by Eq. 3.20. To compute $O(\hat{\boldsymbol{m}}_{e,\text{upd}}^{(n)})$, requires the reservoir predictions for the history-matching period only. However, because the model $\boldsymbol{m}_{e,\text{upd}}^{(n)}$ may be added to the training set, when running the reservoir simulator with $\boldsymbol{m}_{e,\text{upd}}^{(n)}$, we needed to extend the reservoir simulation total time to include the forecast period.

When the performance of the quadratic approximation $q_e^{(n)}(\delta\hat{\boldsymbol{m}}_e^{(n)})$ is satisfactory (see Appendix B), we update the current estimate of the minimum as

$$\hat{\boldsymbol{m}}_e^{(n+1)} = \hat{\boldsymbol{m}}_{e,\text{upd}}^{(n)} = \hat{\boldsymbol{m}}_e^{(n)} + \delta\hat{\boldsymbol{m}}_e^{(n)*} \qquad \text{for} \quad e = 1, 2, \ldots, N_e \,. \tag{3.23}$$

Otherwise, we repeat the trust region minimization using the same current estimate, i.e., $\hat{\boldsymbol{m}}_e^{(n+1)} = \hat{\boldsymbol{m}}_e^{(n)}$, with a reduced the trust region radius (see Appendix B). Nevertheless, we add $\hat{\boldsymbol{m}}_{e,\text{upd}}^{(n)}$ to the training set. Notice that to add a model to the training set, we need to run the reservoir simulator. However, for $\hat{\boldsymbol{m}}_{e,\text{upd}}^{(n)}$ we already have the reservoir simulation results, which were computed when calculating $O(\hat{\boldsymbol{m}}_{e,\text{upd}}^{(n)})$, as discussed above. Thus, no extra reservoir simulation run is indeed required.

As one can see, the training set is updated at each iteration of the minimization

loop. The training examples added to the training set at each iteration are expected to come from regions of increasing probability or at least regions near the modes, given that the minimization step was able to improve the minimum estimate. Consequently, the proposed methodology adapts the training set to the regions around the modes of the posterior pdf. At the end of the minimization process, it is expected that a LS-SVR proxy model trained with the final training set will provide good predictions around the modes of the posterior pdf. Hence, it is expected that the resulting LS-SVR proxy model is a suitable proxy to replace the true reservoir simulator during the MCMC sampling procedure.

As presented in Appendix A, the training procedure for the LS-SVR proxy model requires the computation and inversion of a square matrix with dimension equal to the size of the corresponding training set. Hence, large training sets increase the computational cost of training the LS-SVR proxy model. To avoid an excessively large training set, we adopt a minimum distance criterion to decide when update the training set. The underlying idea is that we wish to avoid adding to the training set points that are too close to those already in the current training set, as such points do not aggregate new information when training the LS-SVR proxy model. To accomplish this, when a new model $\boldsymbol{m}_{e,\text{upd}}^{(n)}$ is obtained, to decide whether it is to be added to the training set, we first compute

$$d_{\min}^{*} = \underset{\boldsymbol{m}_k \in \text{TS}}{\arg\min} \left\| \boldsymbol{m}_{e,\text{upd}}^{(n)} - \boldsymbol{m}_k \right\|_2 . \tag{3.24}$$

In Eq. 3.24, $\boldsymbol{m}_k$, for $k = 1, 2, \ldots, N_t$, represents the models which are already in the current training set. Consequently, $d_{\min}^{*}$ represents the distance between $\boldsymbol{m}_{e,\text{upd}}^{(n)}$ and the model in the training set which is the closest to $\boldsymbol{m}_{e,\text{upd}}^{(n)}$. Let us denote this closest model in the training set as $\boldsymbol{m}_k^{*}$. Then, whenever

$$\frac{d_{\min}^{*}}{\sqrt{N_m}} > d_{\min} , \tag{3.25}$$

where $d_{\min}$ represents a minimum distance defined by the user, we add $\boldsymbol{m}_{e,\text{upd}}^{(n)}$ to the current training set. Notice that we choose to normalize the distance by the square root of the dimension of $\boldsymbol{m}$, i.e., square root of N_m, as in Rafiee and Reynolds [95]. Additionally, for

the cases where

$$\frac{d^*_{\min}}{\sqrt{N_m}} \leq d_{\min} \, , \tag{3.26}$$

but the model $\boldsymbol{m}^{(n)}_{e,\mathrm{upd}}$ provides a value of the objective function less than the value obtained for $\boldsymbol{m}^*_k$, we opt to replace the training example which contains $\boldsymbol{m}^*_k$ by the training example with $\boldsymbol{m}^{(n)}_{e,\mathrm{upd}}$ in the current training set, since we believe that a point which provides lower value of the objective function represents a better option to use in the training set.

Whenever a minimization problem converges, we remove the corresponding minimization problem from the minimization loop, and we continue to iterate for the remaining problems. We adopt four criteria to control the convergence of the minimization problems. The first convergence criterion is related to the trust region radius. We regard a minimization problem as converged whenever the trust region radius becomes less than a minimum threshold, i.e., when $\delta^{(n)}_e < \delta_{\min}$, with $\delta_{\min}$ defined by the user. The reason is that, since the trust region radius is reduced whenever the quadratic approximation does not represent a good approximation of the objective function, the fact that the trust region radius becomes too small is an indication that the quadratic approximation is unable to correctly represent the objective function for that particular minimization problem. Hence, whenever $\delta^{(n)}_e < \delta_{\min}$, we consider the corresponding eth minimization problem as converged, although the current estimate of the minimum, for this particular minimization problem, probably does not represent a good minimum of the objective function.

The second convergence criterion is related to the magnitude of the objective function value. We define the normalized objective function, $O_N(\hat{\boldsymbol{m}}^*)$, as the double of the value of the objective function computed at $\hat{\boldsymbol{m}}^*$ and divided by the total number of observed data. For any reasonable minimum $\hat{\boldsymbol{m}}_{\min}$ of the objective function, one expects that [89]

$$O_N(\hat{\boldsymbol{m}}_{\min}) \equiv \frac{2\, O(\hat{\boldsymbol{m}}_{\min})}{N_{d_h}} \leq 1 + 5\sqrt{\frac{2}{N_{d_h}}} \, . \tag{3.27}$$

Hence, we regard a minimization problem as converged whenever $O_N(\hat{\boldsymbol{m}}^{(n+1)}_e)$ satisfies the condition given by Eq. 3.27.

The third convergence criterion is related to the improvement in the value of the objective function. First we compute the relative change in the objective function value, at the nth iteration for the eth minimization problem, as (see Eq. B.22)

$$\rho_{e,\text{stop}}^{(n)} \equiv \frac{\text{abs}\left[O(\hat{\boldsymbol{m}}_e^{(n+1)}) - O(\hat{\boldsymbol{m}}_e^{(n)})\right]}{\text{abs}\left[O(\hat{\boldsymbol{m}}_e^{(n+1)})\right]} \,. \tag{3.28}$$

In Eq. 3.28, "abs[$\cdot$]" represents the absolute value function. We regard the corresponding minimization problem as converged whenever $\rho_{e,\text{stop}}^{(n)} \leq \epsilon_{\min}$, where the precision parameter $\epsilon_{\min}$ is defined by the user.

Finally, the fourth stop criterion is regarded to the norm of the model update. When the norm of the model update $\delta\hat{\boldsymbol{m}}_e^{(n)*}$ becomes less than a given threshold, we consider the corresponding minimization problem as converged, i.e., when $\|\delta\hat{\boldsymbol{m}}_e^{(n)*}\|_2 < \epsilon_m$, with the precision ϵ_m defined by the user. The idea here is that too small update indicates that the trust region algorithm already reached a local minimum, therefore, no further improvement is expected.

When any of the four convergence criteria is satisfied, we consider the corresponding minimization problem as converged. We stop the minimization loop when a given percentage, p_{conv}, of all N_e minimization problems has converged. We adopt the percentage p_{conv} since some of the minimization problems may take too many iterations in order to converge and we do not need all N_e modes in order to construct the GMM approximation, as we discuss later in this dissertation.

As discussed earlier, the computational cost of training a LS-SVR proxy model increases with the training set size. We already use the minimal distance to update the training set in an attempt to avoid excessively large training sets. However, depending on the values of the objective function for the initial guesses, several iterations may be required for the minimization loop to converge. The training examples which provide high values of the objective function are important in the initial iterations, since one needs a LS-SVR proxy that is able to provide a reliable sensitivity matrix for these initial guesses with high objective

function value. However, after some iterations are conducted, the training examples which provide high objective function values represent an unnecessary computational burden. To circumvent this issue, when the training set contains more than N_{cut} training examples, with N_{cut} defined by the user, we propose to remove from the current training set the training examples which have normalized objective function values greater than a given threshold $O_{N,\text{cut}}$. Eliminating the training examples which provide high normalized objective function values improves the computational efficiency of the minimization loop and does not affect the quality of the resulting LS-SVR proxy model for the MCMC sampling procedure. The reason for this latter statement is that the posterior pdf computed for this high normalized objective function values represent regions of very low probability, i.e., those models represent models with very low probability of occurrence. The details of the minimization process are given in Algorithm 3.1, where r_{conv} represents the number of minimization problems that still running, and $O_{N\text{cut}}$ and N_{cut} are used to reduce the total training set size during the minimization process, as discussed earlier, also, γ and σ represent the LS-SVR parameters, selected as described earlier.

At the end of the minimization process, the resulting training set is used to train a final LS-SVR proxy model. This final LS-SVR proxy model is used to replace the reservoir simulator in the next step, i.e., during the MCMC sampling procedure. It is worth mentioning here that, before training the final LS-SVR, the final training set can be further reduced to improve the computational cost. Depending on the results of the minimization process, we recommend reducing the size of the training set using a more restrictive normalized objective function threshold, $O_{N\text{max}}$, defined by the user. As we comment on before, during the MCMC sampling procedure, relatively high values of the objective function represents regions of very low probability. Furthermore, it is expected that the models from those regions are seldom proposed, and once proposed will most probably be rejected. Hence, the impact of a more restrictive normalized objective function threshold will be nil. We do not recommend using the more restrictive threshold during the minimization process because it could affect the ability of the proxy to provide good search directions for some of the minimization problems

Algorithm 3.1: Find modes of the posterior pdf and construct the training set

1. Set N_e, N_a, $d_{\min}$, ϵ_m, $\epsilon_{\min}$, N_{cut}, $O_{N\text{cut}}$, p_{conv} and the LS-SVR parameters γ and σ. Set the trust region radii $\delta_{\min}$ and $\delta_{\max}$ (see Algorithm B.1).
2. For $e = 1, \ldots, N_e$: sample an initial guesses $\boldsymbol{m}_e^{(1)}$ from the prior pdf and convert it to $\hat{\boldsymbol{m}}_e^{(1)}$ using Eq. 2.11, set the initial trust region radius $\delta_e^{(1)}$.
3. Sample N_a models from the prior pdf. Set $N_t = N_e + N_a$. For $k = 1, \ldots, N_t$: run the reservoir simulator for $\boldsymbol{m}_k$ to compute the corresponding output $\boldsymbol{d}_k$, save the training example $(\boldsymbol{m}_k, \boldsymbol{d}_k)$ in the initial training set **TS**.
4. Set $r_{\text{conv}} = 0$ and $n = 1$
5. **While** $(r_{\text{conv}} < p_{\text{conv}} \times N_e)$

   - **If** $(N_t > N_{\text{cut}})$
     - For $k = 1, \ldots, N_t$: if $O(\boldsymbol{m}_k) > O_{N\text{cut}}$, **REMOVE** $(\boldsymbol{m}_k, \boldsymbol{d}_k)$ from the **TS**.

     End If
   - Train a LS-SVR proxy model using the available **TS**.
   - **For** $(e = 1$ to $N_e - r_{\text{conv}})$
     - Compute the dimensionless sensitivity matrix for $\hat{\boldsymbol{m}}_e^{(n)}$ using Eq. 2.17 and the available LS-SVR proxy model.
     - From $\hat{\boldsymbol{m}}_e^{(n)}$ and $\delta_e^{(n)}$, solve the trust-region minimization sub-problem of Eqs. 3.17 and 3.18 to compute $\delta\hat{\boldsymbol{m}}_e^{(n)*}$ (see Appendix B).
     - Compute $\hat{\boldsymbol{m}}_{e,\text{upd}}^{(n)}$ using Eq. 3.21. Compute $\boldsymbol{m}_{e,\text{upd}}^{(n)}$ using Eq. 3.20 and run the reservoir simulator to compute the corresponding output $\boldsymbol{d}_{e,\text{upd}}^{(n)}$. Compute $\rho_e^{(n)}(\delta\hat{\boldsymbol{m}}_e^{(n)*})$ using Eq. 3.22. Compute $\rho_{e,\text{stop}}^{(n)}$ using Eq. 3.28.
     - Determine $\hat{\boldsymbol{m}}_e^{(n+1)}$ and $\delta_e^{(n+1)}$ (see Algorithms B.1 and B.2).
     - Compute $d_{\min}^*$ for $\boldsymbol{m}_{e,\text{upd}}^{(n)}$ using Eq. 3.24, and also find the training example $(\boldsymbol{m}_{k,\text{close}}, \boldsymbol{d}_{k,\text{close}})$ in the **TS** which is the closest to $\boldsymbol{m}_{e,\text{upd}}^{(n)}$:
       If $(d_{\min}^* > d_{\min})$
       - ▷ **ADD** the training example $(\boldsymbol{m}_{e,\text{upd}}^{(n)}, \boldsymbol{d}_{e,\text{upd}}^{(n)})$ to the **TS**.
       - ▷ Set $N_t = N_t + 1$.

       Else If $(\, O(\boldsymbol{m}_{k,\text{close}}) > O(\boldsymbol{m}_{e,\text{upd}}^{(n)})\,)$
       - ▷ **REPLACE** $(\boldsymbol{m}_{k,\text{close}}, \boldsymbol{d}_{k,\text{close}})$ by $(\boldsymbol{m}_{e,\text{upd}}^{(n)}, \boldsymbol{d}_{e,\text{upd}}^{(n)})$ in the **TS**.

       End If
     - **If** $(\, \rho_{e,\text{stop}}^{(n)} \le \epsilon_{\min}$ or $\|\delta\hat{\boldsymbol{m}}_e^{(n)*}\|_2 < \epsilon_m$ or $\delta_e^{(n+1)} < \delta_{\min}$ or $O_N(\hat{\boldsymbol{m}}_e^{(n+1)})$ satisfies Eq. 3.27)
       - ▷ Set $r_{\text{conv}} = r_{\text{conv}} + 1$, **REMOVE** the current minimization problem from the minimization loop and reorder the remaining (non-converged) minimization problems from 1 to $N_e - r_{\text{conv}}$.

       End If

     End For
   - Set $n = n + 1$.

being conducted.

Additionally, it is worthwhile to mention that different approaches for further reducing the training set could also be investigated, for example, to eliminate points in the training set unnecessarily concentrate in the same region. Here we propose a simple procedure. After the minimization loop converges, whenever the size of the final training set is considerable, we adopt a normalized objective function threshold to select some representative converged minima from all the converged minimization problems. Those minima are used as seed models to generate the final training set. After selecting the seed models, we run a loop comparing the models in the training set one by one with the seed models. Whenever the model is to close to the seed models, we remove the model. Conversely, whenever the model is far from the models in the seed set, we add the model to the seed set. At the end, we use the models as the final training set to train the final LS-SVR proxy model. To compare the models from the training set with the models in the seed set, we use the same approach as in Eq. 3.24, however with a less restrictive value than $d_{\min}$ which was use as a threshold to add models to the training set during the minimization loop.

As a final note, it is worth stressing here that every time a given model $\boldsymbol{m}_k$ is added to the training set, one must run the forward model (the reservoir simulator) to compute the respective predictions $\boldsymbol{d}_k$. Our LS-SVR does not incorporate time as an input value, which means that we need to run the reservoir simulator up to any time we are going to consider. When minimizing the objective function (Eq. 2.13) corresponding to the history-matching period, we only need the predictions corresponding to the observed data period. However, as the final aim is to quantify the uncertainty for predictions of future reservoir performance, for every model added to the training set, we extend the corresponding reservoir simulator to include the forecasting period, and the LS-SVR model is trained to reproduce all the desired production history, including both the history-matching and the forecasting periods.

3.2.2 Second Step: MCMC Sampling Procedure

As discussed earlier, Li and Reynolds [69] suggested the use of a GMM approximation

of the posterior pdf as the proposal distribution in a Metropolis-Hastings MCMC sampling framework. A GMM is a probability distribution which is formed by a weighted summation of a certain number of Gaussian distributions [100]. Denoting the GMM probability distribution by $\pi_{\text{GMM}}(\boldsymbol{m})$, one gets

$$\pi_{\text{GMM}}(\boldsymbol{m}) = \sum_{\ell=1}^{N_g} w_\ell \, \mathcal{N}(\boldsymbol{m}_\ell, C_{M\ell}) \, . \tag{3.29}$$

In Eq. 3.29, N_g represents the selected number of Gaussian distributions, $\mathcal{N}(\boldsymbol{m}_\ell, C_{M\ell})$ denotes a Gaussian distribution with mean $\boldsymbol{m}_\ell$ and corresponding covariance matrix $C_{M\ell}$, and $w_\ell > 0$, for $\ell = 1, 2, \ldots, N_g$, represents the weight of each Gaussian distribution. In order for $\pi_{\text{GMM}}(\boldsymbol{m})$ of Eq. 3.29 represent a proper probability distribution, the constraint

$$\sum_{\ell=1}^{N_g} w_\ell = 1 \tag{3.30}$$

needs to be satisfied.

Following Li and Reynolds [69], we build the GMM approximation centered on some modes of the posterior pdf. As suggested by Li and Reynolds [69], we cluster the modes found in the minimization process into N_c clusters. By clustering these modes, one avoids having essentially the same mode represented by more than one Gaussian in the GMM approximation. Not all converged minimization problems solutions are used in the clustering procedure, we only select the representative ones by using a normalized objective function threshold. Therefore, from the N_e minimization problems solutions, we select the respective modes which provide normalized objective function less than a given threshold $O_{N,\text{cluster}}$. Although one could use the constraint presented in Eq. 3.27 to define the threshold $O_{N,\text{cluster}}$, we believe that including more models in the clustering process results in a GMM which better approximates the posterior pdf. Therefore, we recommend that one use a threshold $O_{N,\text{cluster}}$ less restrictive than the constraint of Eq. 3.27.

We cluster the selected modes into N_c clusters using the k-medoids clustering algorithm proposed by Kaufman and Rousseeuw [63]. The k-medoids proceeds by dividing all

points in a given set into a predetermined number of clusters. Then, for each cluster, the algorithm selects the point which is the closest to the center of the cluster. Those selected points are the medoids, which are used to represent the clusters. The total number of clusters are predefined by the user, which could be consider a drawback of the method. Nevertheless, for the results presented in this dissertation, we follow Li and Reynolds [69] and adopt a total number of clusters given by $N_c = 25$. We denote the resulting clusters as $\boldsymbol{m}_c^*$, for $c = 1, 2, \ldots, N_c$. Thus, each cluster $\boldsymbol{m}_c^*$ represents the mean of one Gaussian in the GMM approximation, i.e., $N_g = N_c$ in Eq. 3.29. The respective corresponding covariance matrix for each Gaussian in the GMM approximation is given by the inverse Hessian of the objective function [69]. Each Hessian is computed for the respective cluster $\boldsymbol{m}_c^*$, for $c = 1, 2, \ldots, N_c$, using Eq. 2.9. Consequently, denoting by $C_{M,c}^*$ the covariance matrix corresponding to the cluster $\boldsymbol{m}_c^*$, one obtains

$$C_{M,c}^* \equiv \left[\mathcal{H}(\boldsymbol{m}_c^*)\right]^{-1} = \left[C_M^{-1} + G_{\text{app}}(\boldsymbol{m}_c^*)^T C_D^{-1} G_{\text{app}}(\boldsymbol{m}_c^*)\right]^{-1}, \tag{3.31}$$

for $c = 1, 2, \ldots, N_c$. In Eq. 3.31, the required approximate sensitivity matrix $G_{\text{app}}(\boldsymbol{m}_c^*)$ is computed using Eq. 3.19 for $\boldsymbol{m}_c^*$ and the final LS-SVR proxy model which is trained at the end of the minimization process.

Using the GMM approximation of Eq. 3.29 for the computed mean and covariance matrix given by the clusters, as described above, our proposal distribution $q(\boldsymbol{m}_s|\,\boldsymbol{m}_\ell)$ for the Metropolis-Hastings algorithm is given by

$$\begin{aligned} q(\boldsymbol{m}_s|\,\boldsymbol{m}_\ell) &= \pi_{\text{GMM}}(\boldsymbol{m}_s) \\ &= \sum_{c=1}^{N_c} \frac{w_c}{\sqrt{(2\pi)^{N_m}|C_{M,c}^*|}} \exp\left(-\frac{1}{2}(\boldsymbol{m}_s - \boldsymbol{m}_c^*)^T\, C_{M,c}^{*-1}\,(\boldsymbol{m}_s - \boldsymbol{m}_c^*)\right). \end{aligned} \tag{3.32}$$

Following the recommendation of Rafiee and Reynolds [95], in Eq. 3.32 we adopt $w_c = 1/N_c$, for $c = 1, 2, \ldots, N_c$.

The proposal distribution $q(\boldsymbol{m}_s|\,\boldsymbol{m}_\ell)$ of Eq. 3.32 gives the probability of proposing

a transition from the state $\boldsymbol{m}_\ell$ to a new state $\boldsymbol{m}_s$ in the chain. As pointed out by Li and Reynolds [69], with the proposal distribution of Eq. 3.32, the probability of proposing the new state $\boldsymbol{m}_s$ is independent of the current state $\boldsymbol{m}_\ell$, i.e., $q(\boldsymbol{m}_s|\,\boldsymbol{m}_\ell)$ does not depend on $\boldsymbol{m}_\ell$. As a consequence, the proposal distribution promotes good mixing in the chain, which tends to avoid having the chain trapped near the same mode for a large number of states. This is a desirable characteristic when sampling from a complex non-Gaussian pdf, such as the posterior pdf for reservoir simulator parameters.

The results presented by Li and Reynolds [69] and Rafiee and Reynolds [95] showed that the GMM proposal distribution of Eq. 3.32 indeed accelerates the convergence of the corresponding Markov chain. As discussed before, to evaluate the Metropolis-Hastings acceptance probability one reservoir simulation run is required (see Algorithm 2.2). Consequently, the Li and Reynolds [69] and Rafiee and Reynolds [95] approaches result in a considerable reduction in the total number of required reservoir simulation runs. However, they found that tens of thousands of reservoir simulations runs were still required to properly investigate the target pdf, i.e., the posterior pdf, when sampling with MCMC.

The critically important difference between the methodology developed in this research and those of Li and Reynolds [69] and Rafiee and Reynolds [95] is that we use the final LS-SVR model to compute the Metropolis-Hastings acceptance probability. As a consequence, we do not need to run the reservoir simulator during the MCMC sampling procedure. However, by using the LS-SVR proxy model, instead of sampling the true posterior pdf $\pi(\boldsymbol{m})$ of Eq. 2.6, we are sampling from the following approximation, $\pi_{\text{app}}(\boldsymbol{m})$, of the posterior pdf

$$\pi_{\text{app}}(\boldsymbol{m}) = a_d \, \exp\big[- O_{\text{app}}(\boldsymbol{m})\big] \, . \tag{3.33}$$

In Eq. 3.33, a_d represents the normalizing constant, and $O_{\text{app}}(\boldsymbol{m})$ represents the approximated objective function which is obtained by replacing the reservoir simulator predictions

with the LS-SVR proxy predictions. Therefore

$$O_{\text{app}}(\boldsymbol{m}) = \frac{1}{2}(\boldsymbol{m} - \boldsymbol{m}_{\text{pr}})^T C_M^{-1}(\boldsymbol{m} - \boldsymbol{m}_{\text{pr}}) + \frac{1}{2}\Big(\hat{g}_h(\boldsymbol{m}) - \boldsymbol{d}_{\text{obs}}\Big)^T C_D^{-1}\Big(\hat{g}_h(\boldsymbol{m}) - \boldsymbol{d}_{\text{obs}}\Big)\,. \quad (3.34)$$

In Eq. 3.34, $\hat{g}_h(\boldsymbol{m}) = [\hat{g}_1(\boldsymbol{m}), \hat{g}_2(\boldsymbol{m}), \ldots, \hat{g}_{N_{d_h}}(\boldsymbol{m})]^T$ denotes the column vector of predictions from the LS-SVR proxy model which correspond to the history matching period. The entries $\hat{g}_i(\boldsymbol{m})$, for $i = 1, 2, \ldots, N_{d_h}$, are computed using Eq. 3.12 for the final LS-SVR proxy model.

As evaluating the analytical expression for LS-SVR proxy is significantly faster than running the real reservoir simulator, the computational cost of running a long Markov chain is feasible when using the proxy predictions to compute the Metropolis-Hastings acceptance probability. Algorithm 3.2 gives the proposed MCMC sampling methodology.

As discussed earlier, the performance of many Metropolis-Hastings MCMC algorithms is greatly dependent on the choice of the proposal distribution. For many practical applications it is difficult to correctly tune the selected proposal distribution in order to deliver an efficient MCMC algorithm. Several previous works have proposed to automatically tune the proposal while the MCMC sampling is conducted. Here, we follow Rafiee and Reynolds [95] and use the covariance matrix adaptation as presented in Eq. 2.65. As discussed before, we chose the parameter μ_γ (see the discussion after Eq. 2.65) such that the resulting Metropolis-Hastings MCMC sampling algorithm delivers an acceptance rate between 0.20 and 0.50 [104, 101].

To monitor the convergence of the resulting Markov chain, we follow Li and Reynolds [69] and adopt the MPSRF procedure as described in Sub-Section 2.4.1.

3.3 Least-Squares Support Vector Regression and Markov Chain Monte Carlo for Large Scale Problems

For Li and Reynolds [69] and Rafiee and Reynolds [95] approach, which uses the reservoir simulator, using a MCMC sampling framework for uncertainty quantification is

Algorithm 3.2: Proposed Metropolis-Hastings MCMC algorithm

1. Cluster the found modes into N_c clusters and build a GMM approximation of the posterior pdf to use as proposal distribution (see Eq. 3.32).

2. Set the state counter $s = 0$ and choose the initial state $\boldsymbol{m}_0$ in the chain by sampling the GMM approximation of Eq. 3.32.

3. Propose a candidate to new state in the chain, $\widetilde{\boldsymbol{m}}_{s+1}$, by sampling the GMM proposal distribution of Eq. 3.32.

4. Evaluate the Metropolis-Hastings acceptance probability using the approximate objective function $\pi_{\text{app}}(\boldsymbol{m})$ of Eq. 3.33

$$\alpha(\boldsymbol{m}_s,\, \widetilde{\boldsymbol{m}}_{s+1}) = \min\left\{1,\, \frac{\pi_{\text{app}}(\widetilde{\boldsymbol{m}}_{s+1})\; \pi_{\text{GMM}}(\boldsymbol{m}_s)}{\pi_{\text{app}}(\boldsymbol{m}_s)\; \pi_{\text{GMM}}(\widetilde{\boldsymbol{m}}_{s+1})}\right\}. \tag{3.35}$$

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

6. If $u < \alpha(\boldsymbol{m}_s,\, \widetilde{\boldsymbol{m}}_{s+1})$, the proposed candidate is accepted as the new state in the chain, i.e., set $\boldsymbol{m}_{s+1} = \widetilde{\boldsymbol{m}}_{s+1}$. Otherwise, the proposed candidate is rejected and the current state is repeated in the chain, i.e., set $\boldsymbol{m}_{s+1} = \boldsymbol{m}_s$.

7. Increase the chain counter by making $s = s + 1$ and return to Step 3 until the desired chain length is achieved.

virtually infeasible for large scale problems. Here, large scale problems refer to the cases for which the reservoir simulation model is composed of tens of thousands to millions parameters, i.e., the cases for which N_m is large.

Although the methodology presented in this dissertation seemingly allows one to apply MCMC even for large scale problems, direct application of the proposed methodology for large scale problems represent a computational burden for both the LS-SVR training procedure and the minimization process depicted in the previous sections. In particular, large scale problems results in large covariance matrices, which makes it computationally difficult to sample from and evaluate the GMM proposal distribution, besides the high computational cost to adapt the covariance matrices.

To circumvent this issues and handle large scale problems, an order reduction technique is applied. Here, we adopt the principal component analysis (PCA) [137, 110]. The basic idea of the PCA method is to decompose a given matrix into its respective main directions. The decomposition is conducted in such a way that each main direction is related to a principal variance direction. By principal variance directions, we mean the directions which have more influence in the linear transformation represented by the given matrix. Conventionally, the first main direction is related to the highest variance, the second main direction to the second highest variance, and so on. To effectively reduce the order of a problem, the PCA must be able to accurately reconstruct the given matrix by using only few main directions, the ones which are related to the highest variances, i.e., the ones which have more influence in the linear transformation represented by the given matrix.

In our history matching and uncertainty quantification applications, one could apply PCA to the prior covariance matrix C_M (see Eq. 2.7) to reduce the order of the input vector $\boldsymbol{m}$, see, for example, Reynolds et al. [96]. However, the prior covariance matrix may pertain to variables which have vastly different variances. By applying PCA directly to the prior covariance matrix, one could potentially eliminate the dependence on important uncertain quantities, not because they have low influence in the given problem, but instead because of the different numerical scales. To circumvent this limitation, we advocate to instead apply PCA to the corresponding prior correlation matrix. The correlation matrix is obtained from the covariance matrix when dividing each entry by the product of the corresponding standard deviations. Consequently, all entries of the correlation matrix have numerical values less than or equal to unity, and all diagonal entries are equal to unity.

Denoting the prior correlation matrix as $\widetilde{C}_M$, it follows that

$$\widetilde{C}_M = S_M^{-1}\, C_M\, S_M^{-1} \quad \Rightarrow \quad C_M = S_M\, \widetilde{C}_M\, S_M\,. \tag{3.36}$$

In Eq. 3.36, S_M denotes the standard deviation matrix, which is the diagonal matrix for which the jth entry of the diagonal is given by the standard deviation of the jth parameter,

i.e., the standard deviation for the jth entry of the vector of reservoir model parameters $\boldsymbol{m}$, and S_M^{-1} denotes the inverse of S_M. Since S_M is a diagonal matrix, it is symmetric. Since C_M is a symmetric positive definite matrix, it is easy to verify that $\widetilde{C}_M$, defined in Eq. 3.36, is also a symmetric positive definite matrix.

To apply PCA, we start by decomposing the correlation matrix $\widetilde{C}_M$ using the singular value decomposition (SVD) [46]

$$\widetilde{C}_M = U\, W\, V^T\,. \tag{3.37}$$

In Eq. 3.37, the $N_m \times N_m$ matrix U denotes the orthogonal matrix of left singular vectors of $\widetilde{C}_M$, i.e., the columns of U represent the unit length left singular vectors $\boldsymbol{u}_j$, for $j = 1, 2, \ldots, N_m$, of the correlation matrix $\widetilde{C}_M$. Similarly, the $N_m \times N_m$ matrix V denotes the orthogonal matrix of right singular vectors of $\widetilde{C}_M$, i.e., the columns of V represent the unit length right singular vectors $\boldsymbol{v}_j$, for $j = 1, 2, \ldots, N_m$. Since $\widetilde{C}_M$ is symmetric positive definite, it holds that $U \equiv V$, and the columns of U also represent the eigenvectors of the symmetric positive definite matrix $\widetilde{C}_M$. The matrix $U \equiv V$ is orthogonal, i.e.,

$$U\, U^T = U^T\, U = I_{N_m}\,, \tag{3.38}$$

where I_{N_m} denotes the $N_m \times N_m$ identity matrix.

Also in Eq. 3.37, the $N_m \times N_m$ diagonal matrix W has the jth singular value, denoted by w_j, of the correlation matrix $\widetilde{C}_M$, as its jth diagonal entry. Again, since $\widetilde{C}_M$ is symmetric positive definite matrix, w_j also represents the jth eigenvalue of the matrix $\widetilde{C}_M$. The diagonal elements of the matrix W are ordered such that $w_1 \geq w_2 \geq \ldots \geq w_{N_m}$. The magnitude of the singular values define the principal variance directions, i.e., the highest singular value, i.e., the first element in the diagonal of matrix W, corresponds to the highest principal variance direction. In this case, the principal variance direction points in the direction of the first left singular vector, i.e., the direction given by the first column of the matrix U. The second highest principal variance direction corresponds to the second highest singular value, i.e., the second element in the diagonal of matrix W. Again, the second principal variance

direction points in the direction given by the second column of matrix U, and so on.

Using Eq. 3.38, one can manipulate Eq. 3.37 to obtain

$$\widetilde{C}_M = U\, W\, U^T = U\, W^{1/2}\, W^{T/2}\, U^T = U\, W^{1/2}\, U^T\, U\, W^{T/2}\, U^T \,. \tag{3.39}$$

In Eq. 3.39, the $N_m \times N_m$ diagonal matrix $W^{1/2}$ represents the square root of the matrix W, and $W^{T/2}$ represents the transpose of $W^{1/2}$. Since W is a positive definite diagonal matrix, $W^{1/2} = W^{T/2}$. Furthermore, the jth entry of the diagonal of $W^{1/2}$ is given by $w_j^{1/2}$, for $j = 1, 2, \ldots, N_m$.

To apply the PCA method, we reconstruct the correlation matrix $\widetilde{C}_M$ using only the N_v directions which are related to the highest variance. To accomplish this, we keep only the first N_v columns of the matrix U and only the first N_v diagonal elements of the matrix W, i.e.,

$$\widetilde{C}_M = U_{N_v}\, W_{N_v}\, U_{N_v}^T = U_{N_v}\, W_{N_v}^{1/2}\, W_{N_v}^{T/2}\, U_{N_v}^T = U_{N_v}\, W_{N_v}^{1/2}\, U_{N_v}^T\, U_{N_v}\, W_{N_v}^{T/2}\, U_{N_v}^T \,. \tag{3.40}$$

In Eq. 3.40, U_{N_v} represents the $N_m \times N_v$ matrix formed by the first N_v columns of the matrix U. Similarly, W_{N_v} represents the $N_v \times N_v$ diagonal matrix which is composed of the first N_v elements of the diagonal of the matrix W. It is well known that

$$U_{N_v}^T\, U_{N_v} = I_{N_v} \,, \tag{3.41}$$

where I_{N_v} denotes the $N_v \times N_v$ identity matrix, but when $N_v < N_m$, normally $U_{N_v} U_{N_v}^T = I_{N_m}$.

As will be clear in the following development, N_v represents the dimension of the reduced order problem. Usually, one determines N_v based on some percentage p_{N_v} of the total variance, which is defined as

$$p_{N_v} = \frac{\sum_{j=1}^{N_v} w_j}{\sum_{j=1}^{N_m} w_j} \,. \tag{3.42}$$

Then one determines N_v as the smallest dimension for which p_{N_v} is greater than an user given threshold.

Combining Eqs. 3.36 and 3.40, gives

$$\begin{aligned} C_M &= S_M\, U_{N_v}\, W_{N_v}^{1/2}\, U_{N_v}^T\, U_{N_v}\, W_{N_v}^{T/2}\, U_{N_v}^T\, S_M^T \\ &= \left(S_M\, U_{N_v}\, W_{N_v}^{1/2}\, U_{N_v}^T\right)\, \left(S_M\, U_{N_v}\, W_{N_v}^{1/2}\, U_{N_v}^T\right)^T \ . \end{aligned} \tag{3.43}$$

Conventionally, to determine a suitable square root of the covariance matrix, normally the following properties are enforced:

$$C_M = C_M^{1/2}\, C_M^{T/2} \quad \text{and} \quad C_M^{-1} = C_M^{-T/2}\, C_M^{-1/2} \ . \tag{3.44}$$

In Eq. 3.44, $C_M^{1/2}$ denotes the square root of the matrix C_M and $C_M^{T/2}$ denotes the transpose of $C_M^{1/2}$, while $C_M^{-1/2}$ denotes the inverse of $C_M^{1/2}$ and $C_M^{-T/2}$ denotes the transpose of $C_M^{-1/2}$ which is also the inverse of $C_M^{T/2}$.

From Eq. 3.43, a suitable square root of C_M is given by

$$C_M^{1/2} = S_M\, U_{N_v}\, W_{N_v}^{1/2}\, U_{N_v}^T \ , \tag{3.45}$$

which implies that its pseudo-inverse is given by

$$C_M^{-1/2} = U_{N_v}\, W_{N_v}^{-1/2}\, U_{N_v}^T\, S_M^{-1} \ , \tag{3.46}$$

where $C_M^{-1/2}$ represents the Moore-Penrose [78, 90] pseudo-inverse of $C_M^{1/2}$. It is a pseudo inverse because, although Eq. 3.41 holds for $U_{N_v}^T\, U_{N_v}$, in general, $U_{N_v}\, U_{N_v}^T$ is not equal to the identity matrix I_{N_m} for the case where $N_v < N_m$.

Based on Eqs. 3.45 and 3.46, we adopt the following linear transformation of the

reservoir parameter vector $\boldsymbol{m}$

$$\boldsymbol{z} = W_{N_v}^{-1/2}\, U_{N_v}^T\, S_M^{-1}\, (\boldsymbol{m} - \boldsymbol{m}_{\rm pr})\,. \tag{3.47}$$

Notice that $\boldsymbol{z}$ is a N_v-dimensional column vector. Therefore, the original dimension of the problem, which is given by the dimension of the vector $\boldsymbol{m}$, is effectively reduced to N_v. Furthermore, since $\boldsymbol{m} \sim \mathcal{N}(\boldsymbol{m}_{\rm pr}, C_M)$, it holds that $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, I_{N_v})$, i.e., $\boldsymbol{z}$ follows a Gaussian distribution with zero mean and identity covariance matrix.

With the linear transformation given by Eq. 3.47, the objective function $O(\boldsymbol{m})$ of Eq. 2.7 reduces to

$$O(\boldsymbol{z}) = \frac{1}{2}\boldsymbol{z}^T\,\boldsymbol{z} + \frac{1}{2}\Big(g_h(\boldsymbol{z}) - \boldsymbol{d}_{\rm obs}\Big)^T\, C_D^{-1}\, \Big(g_h(\boldsymbol{z}) - \boldsymbol{d}_{\rm obs}\Big)\,. \tag{3.48}$$

In Eq. 3.48, we consider the reservoir simulator $g_h(\cdot)$ as a function of the input vector $\boldsymbol{z}$, since for any vector $\boldsymbol{m}$, the reduced order vector $\boldsymbol{z}$ can be computed using Eq. 3.47. Conversely, for a given reduced order vector $\boldsymbol{z}$, the corresponding vector $\boldsymbol{m}$ counterpart is computed using

$$\boldsymbol{m} = \boldsymbol{m}_{\rm pr} + S_M\, U_{N_v}\, W_{N_v}^{1/2}\, \boldsymbol{z}\,, \tag{3.49}$$

which is obtained using the pseudo-inverse as described above.

The use of PCA with our proposed methodology is straightforward. Foremost, instead of training the LS-SVR for the reservoir vector of parameters $\boldsymbol{m}$, we train the LS-SVR proxy using its reduced order counterpart $\boldsymbol{z}$. Consequently, the training procedure presented in Appendix A is conducted for the training examples $(\boldsymbol{z}_k,\, \boldsymbol{d}_k)$, for $k = 1,\, 2,\, \ldots\,,\, N_t$, where $\boldsymbol{d}_k = g_h(\boldsymbol{z}_k)$. It can be verified that the reduction in order also represents a reduction in the computational cost of the training procedure. Although most of the computational cost for training a LS-SVR model is due the inversion of the kernel matrix, some computational gain is expected when using the reduced order vector, mainly because the computational cost of computing inner products for the reduced order vector is small.

With the LS-SVR proxy model trained for the reduced order vector $\boldsymbol{z}$, we conduct the minimization process described earlier with the vector $\boldsymbol{m}$ replaced by its reduced order counterpart $\boldsymbol{z}$. Notice that in the objective function, the covariance matrix C_M is replaced by the identity matrix I_{N_v}, which tends to reduce the ill-posedness of the minimization problem and enhances the convergence. Therefore, for large scale problems, the objective function which is indeed minimized is $O(\boldsymbol{z})$ of Eq. 3.48, with the LS-SVR proxy model trained in terms of $\boldsymbol{z}_k$ and $\boldsymbol{d}_k$.

Finally, in the MCMC sampling procedure, the GMM approximation of the posterior pdf is constructed for the reduced order vector $\boldsymbol{z}$. Consequently, the Markov chain is constructed for the vector $\boldsymbol{z}$. For the reduced order case, considerable reduction in the computational burden is also expected during the MCMC sampling procedure when compared with the case where the vector $\boldsymbol{m}$ is used instead. The advantage arises from sampling from a lower order multivariate Gaussian distribution, as well as from adapting a lower order covariance matrix.

It is worth pointing out here that by coupling the LS-SVR proxy with the PCA method, or any other reduced-order method, the forward model used during the MCMC sampling procedure, i.e., the LS-SVR proxy itself, is indeed a reduced order model. The same may not be true for cases where PCA is coupled with the real reservoir simulator, i.e., one can use the reduced order objective function during the minimization process, however, for constructing the Markov chain one would still need to run the full scale reservoir simulation model.

3.4 Improving the Least-Squares Support Vector Regression Predictions

As the results will show in the next section, building the training set in an iterative framework, adapting it to the regions of interest, results in a LS-SVR proxy model sufficiently accurate for the MCMC sampling procedure. However, as discussed earlier, for a given application we adopt the RBF kernel since no clear procedure to select and/or derive a kernel function is available in the literature [112].

In an attempt to enhance the LS-SVR proxy model predictions, we introduce a novel training procedure which tries to assimilate some physical insights of the problem. The proposed approach is very simple. It consists of using some analytical solution, obtained from a simplified geometry, to derive a mapping function. The mapping function is applied to the input vectors before the training procedure is conducted, as depicted next.

As discussed earlier, the LS-SVR adopts a kernel function which represents the inner product of some given input vectors in some high dimensional feature space [18, 112]. The underlying idea is that the relationship between input vectors and corresponding outputs is linear in the feature space [18, 112]. Theoretical results [92, 25, 115] suggests that for any non-linear function, there exist a feature space which makes the relationship between input vectors and corresponding outputs linear. The feature space is defined by selecting a specific kernel function. However, no clear procedure is known to determine such a feature space, i.e., no clear procedure of how to derive a kernel function for a given problem is available. In practice, one adopts some kernel function family, such as the RBF kernel, then tunes some parameters to adjust the chosen kernel function for a given problem. The resulting kernel function does not necessarily represent a feature space where the relationship between input and output vectors is linear, although historically the resulting SVR model gives an acceptable performance for most applications. For the classification problem, [108] present a methodology to incorporate prior knowledge about the data into the resulting SVM by multiplying the input vectors by some pre-processing matrix. Recently, it has become popular to combine multiple available kernels, instead of adopting a specific one [11, 10, 47].

We propose an alternative approach which tries to assimilate physical knowledge from a given problem. The underlying idea is simplistic. Suppose one needs to build a LS-SVR proxy model for a given reservoir simulator $\boldsymbol{d} = g(\boldsymbol{m})$, using the training set

$$T_s = \{(\boldsymbol{m}_k, \boldsymbol{d}_k), \quad k = 1, 2, \ldots, N_t\} . \tag{3.50}$$

As before, in Eq. 3.50, $\boldsymbol{d}_k = g(\boldsymbol{m}_k)$. For cases where the relationship defined by $g(\boldsymbol{m})$ is

very non-linear and complex, the RBF kernel may struggle to deliver a reasonable LS-SVR. However, even for complex reservoirs, a simplified analytical solution may still able to provide some useful insights on the fluid flow dynamics in the porous media. By simplified analytical solutions, we mean solutions for uniform reservoir properties and simplified geometry, as well as a simplified fluid model. Although simplified analytical solutions are able to only roughly reproduce the real reservoir numerical simulator, it is possible to derive from an analytical solution some dependency relationships. That is, one is able to relate how the pressure or flow rates will depend on the permeability, for example.

Here we propose to derive a simplified analytical solution for a given problem. Then, from the analytical solution we derive some one-dimensional dependence relationship, which we denote by $\psi(\cdot)$, between the input properties of the reservoir model and some observed quantity. For example, one could derive a dependence relation between permeability and flow rate. Finally, we use $\psi(\cdot)$ as a mapping function for the entries of the vector $\boldsymbol{m}$. By applying this mapping, we intend to reduce the complexity between the input vector $\boldsymbol{m}$ and the output predictions $\boldsymbol{d}$. Consequently, instead of using the training set given by Eq. 3.50, we propose to train the LS-SVR proxy model using the modified training set

$$T_{s,\text{map}} = \{(\psi(\boldsymbol{m}_k), \boldsymbol{d}_k), \quad k = 1,\, 2,\, \ldots\, ,\, N_t\}\,. \tag{3.51}$$

In Eq. 3.51, the jth entry of the N_m-dimensional vector $\psi(\boldsymbol{m}_k)$ is given by $\psi(m_j)$, for $j = 1,\, 2,\, \ldots\, ,\, N_m$, where m_j represents the jth entry of the vector $\boldsymbol{m}$.

By careful examination of the training procedure presented in Appendix A, one may conclude that the only effect of the mapping presented above is to modify the chosen kernel function. For the RBF kernel adopted here, the kernel function is modified to

$$K(\boldsymbol{m}_k, \boldsymbol{m}) = K\left(\psi(\boldsymbol{m}_k), \psi(\boldsymbol{m})\right) = \exp\left(-\frac{\left\|\psi(\boldsymbol{m}_k) - \psi(\boldsymbol{m})\right\|_2^2}{\sigma^2}\right)\,. \tag{3.52}$$

Therefore, the proposed mapping procedure tries to adapt the kernel function to a particular given problem.

Although the results obtained so far indicates that the simplistic mapping procedure described in this section indeed improves the proxy predictions, it is very hard to generalize this mapping methodology. The main reason is that, for practical applications, both the vectors $\boldsymbol{m}$ and $\boldsymbol{d}$ contain different physical entities as its entries, which makes the derivation of a practical mapping very difficult. Besides, the mapping approach is incompatible with the reducing order techniques, such as PCA. The reduced order vector $\boldsymbol{z}$ is a mathematical entity and does not have a clear physical meaning, hence an analytical solution fails in provide insights about the dependency on the vector $\boldsymbol{z}$.

CHAPTER 4

APPLICATIONS, RESULTS AND DISCUSSIONS

In this chapter we present the application of the proposed methodology for five cases. The first case presented is an 1-D toy problem, which has only one parameter and only one observed datum. As the second case, we present the application to a 1-D reservoir model developed by Emerick and Reynolds [28] to evaluate the performance of different methods for sampling the posterior probability density function (pdf). The third case is a synthetic 2-D reservoir model design by Li [68] to evaluate her two-level MCMC method. The fourth case is the well known 3-D reservoir model PUNQ-S3. Finally, the fifth case is a large scale 3-D reservoir model that we generated based on the available data from the PUNQ-S3 case.

4.1 Application to a Toy Problem.

Although the toy problem presented here is a very simple example, with one parameter and one observed datum, it is relatively easy to compute and visualize the posterior pdf for this simple case. Therefore, it is possible to compare the approximated posterior pdf that we sample with our proposed methodology with the real posterior pdf for this toy problem.

The toy problem presented here is similar to the one designed by Zafari and Reynolds [139]. The one-dimensional forward model is given by

$$d = g(m) = 1.0 - 4.5\left[m - \frac{2\pi}{3}\right]^2 . \tag{4.1}$$

In Eq. 4.1, $m \in \mathbb{R}$ represents the single model parameter. For this case, the prior pdf for the parameter m is assumed Gaussian with mean equal to 2.30 and standard deviation of 0.20, i.e., $m \sim \mathcal{N}(2.30, 0.20^2)$. The true model parameter was sampled from the prior pdf and is equal to $m_{\text{true}} = 1.8836$. To generate a single observed datum, white noise, with standard

deviation equal to 0.10, is added to the prediction obtained for the true model m_{true}. The resulting observed datum is $d_{\text{obs}} = 0.7942$.

For Gaussian prior and Gaussian measurements error distributions, the posterior pdf is given by $\pi(m)$ of Eq. 2.6. As a consequence, the corresponding objective function is given by

$$O(m) = \frac{1}{2}\left[\frac{m - 2.30}{0.20}\right]^2 + \frac{1}{2}\left[\frac{g(m) - d_{\text{obs}}}{0.10}\right]^2 . \tag{4.2}$$

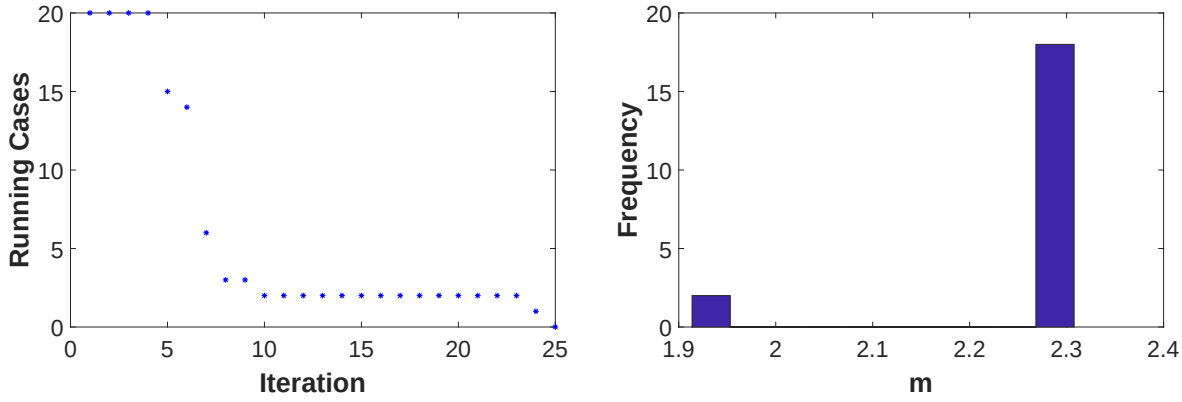
To apply the proposed methodology and construct a GMM approximation for the posterior pdf, we chose to solve $N_e = 20$ minimization problems. The required $N_e = 20$ initial guesses are sampled from the prior Gaussian pdf for m using LHS [74]. An additional $N_a = 60$ points are also sampled from the prior pdf, using LHS, to form an initial training set with $N_t = 80$ training examples.

The minimization loop set up given next is based on the values adopted by Rafiee [93]. The minimum distance to update the training set is selected equal to $d_{\min} = 1.0\text{E}-5$. The trust-region parameters are set equal to $\delta_1^{(1)} = 0.10$, $\delta_{\min} = 1.0\text{E}-5$ and $\delta_{\max} = 0.50$. The convergence criteria are set equal to $\epsilon_{\min} = \epsilon_m = 1.0\text{E}-6$ and $p_{\text{conv}} = 1.0$, which means that all minimization problems have to converge.

The LS-SVR parameters are set equal to $\gamma = 800$ and $\sigma = 0.20$, which means $c_\sigma = 0.20$. As described earlier, this value of c_σ was determined using numerical experiments.

The minimization loop results are presented in Fig. 4.1. As presented in Fig. 4.1(a), the algorithm takes 25 iterations to converge, i.e., for all $N_e = 20$ minimization problems to converge to a local minima. For this toy problem, all the points added to the training set are used to train the final LS-SVR. As shown in Fig. 4.1(b), two distinct modes are found. Note that two minimization problems converge to the mode at $m = 1.9137$ and eighteen minimization problems converge to the second mode at $m = 2.3077$. Those two modes are very close to the actual two modes of the posterior pdf, which can be computed from Eq. 4.2 as $m_1 = 1.9141$ and $m_2 = 2.3077$.

Following the proposed methodology of Chapter 3, we use the modes found to con-



(a) (b)

Figure 4.1: Results of the minimization loop for the toy problem case: (a) number of cases that still running per each iteration, (b) histogram of converged models showing the two distinct modes found.

struct a GMM approximation of the posterior pdf, which results in

$$\pi_{\text{GMM}}(m) = 0.50\,\mathcal{N}(1.9137,\, 0.0032) + 0.50\,\mathcal{N}(2.3077,\, 0.0025)\,; \tag{4.3}$$

note that we use equal weights for each Gaussian as suggested by Rafiee and Reynolds [95].

The variances shown in the GMM approximation of Eq. 4.3 represent the inverse Hessian of the objective function $O(m)$ of Eq. 4.2 at the modes. For this single parameter case, the Hessian represents the second derivative of the objective function. Computing the inverse of the second derivative at the modes found, we obtain 0.0032 and 0.0025, which are precisely the values shown in Eq. 4.3. A comparison between the resulting GMM approximation and the true posterior pdf is presented in Fig. 4.2.

For the MCMC uncertainty quantification step, using the GMM approximation of Eq. 4.3 as proposal distribution, we run our proposed Metropolis-Hastings algorithm for five distinct chains. Each chain starts from a randomly select initial state, which is sampled from the prior pdf. As discussed earlier, the purpose of running five chains is to monitor the convergence using the MPSRF method described earlier. We chose to run each chain with a total length of 50,000 states for this simple toy problem.

The MPSRF convergence criterion for the five chains is presented in Fig. 4.3(a). As

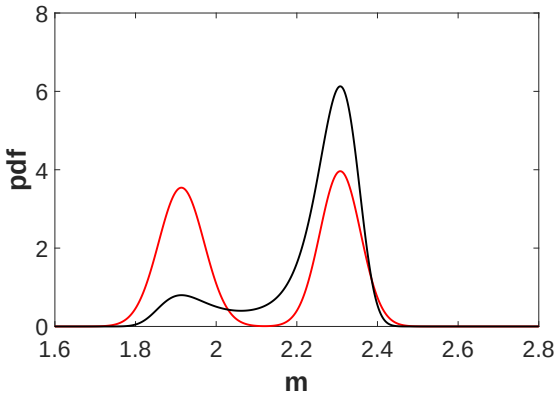


Figure 4.2: Comparison between the GMM approximation of the posterior pdf (red curve) and the real posterior pdf (black curve) for the toy problem case.

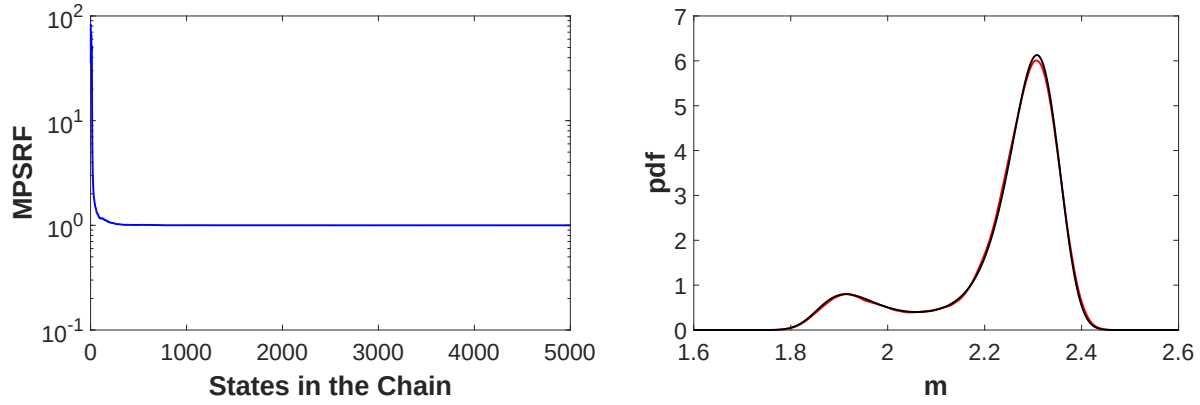
one can observe in Fig. 4.3(a), the MPSRF reaches unity after 400 states are generated, hence, we regard all five chains as converged at state 400. For a bi-modal distribution, convergence after only 400 states are generated, represents very fast convergence when sampling the pdf with MCMC, and this result corroborates the efficiency of Li and Reynolds [69, 68] proposition.

The uncertainty quantification results are presented in Fig. 4.3(b), in which we present a comparison between the true posterior pdf and a pdf constructed using a sample of 20,000 states from each chain, i.e., in a total of 100,000 states. The 20,000 states are randomly selected from each chain after the first 500 states. As one can conclude from Fig. 4.3(b), there is a very good agreement between the true pdf and the pdf reconstructed using samples from all five chains, which supports the applicability of our proposed methodology.

4.2 Application to an One-Dimensional Water-Flooding Model.

This one-dimensional water-flooding reservoir case was designed by Emerick and Reynolds [28] to evaluate the performance of different methods for sampling the posterior pdf. The 1-D reservoir geometry is composed of 31 gridblocks, with dimensions of 50 ft $\times$ 50 ft $\times$ 50 ft for each gridblock. A schematic of this one-dimensional reservoir model is presented in Fig. 4.4, in which we also present the location of wells.

The reservoir properties description are given next. The initial reservoir pressure is



(a) (b)

Figure 4.3: Uncertainty quantification results for the toy problem: (a) convergence of the five chains using the MPSRF method, (b) comparison between true posterior pdf (black curve) and the pdf constructed using a sample of 20,000 states from each chains (blue curve).

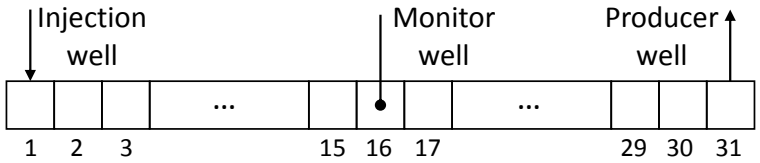


Figure 4.4: One-dimensional water-flooding reservoir model schematic, also presenting the wells locations.

assumed to be constant and equal to 3,500 psi. The porosity of each gridblock is also assumed to be constant and equal to 0.25. Both oil and water viscosity are assumed to be constant and equal to 2.0 cp and 1.0 cp, respectively. The first gridblock contains a water injection well, which operates at a constant bottom-hole pressure of 4,000 psi. The last gridblock contains a production well, which operates at a constant bottom-hole pressure of 3,000 psi.

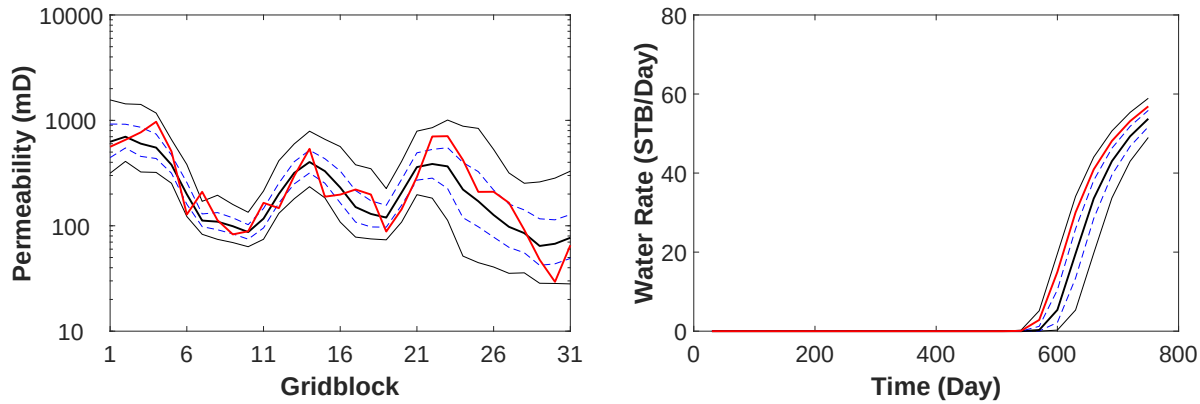
For this one-dimensional water-flooding model, the vector of reservoir model parameters considered for history matching, i.e., the vector $\boldsymbol{m}$, is given by the natural logarithm of each gridblock permeability. Consequently, $N_m = 31$ and $m_j = \ln(k_j)$, for $j = 1, 2, \ldots, N_m$, where k_j represents the permeability of the jth gridblock, and m_j represents the jth entry of the vector $\boldsymbol{m}$. The log-permeability is assumed to follow a multivariate Gaussian distribution with mean $\boldsymbol{m}_{\text{pr}} = [5.0, \ldots, 5.0]^T$, i.e., $m_{\text{pr},j} = 5.0$, for $j = 1, 2, \ldots, N_m$. The variance of each gridblock log-permeability is assumed equal to 1.0. The prior covariance matrix is

the same one used by Emerick and Reynolds [28], which was generated using an exponential covariance function with range set equal to 10 gridblocks.

The true model adopted here is the same one used by Emerick and Reynolds [28], which was randomly sampled from the prior pdf. The dynamic data considered for history matching are the bottom-hole pressure of a monitor well located at the middle of the reservoir, i.e., in gridblock 16. A total of 360 days of operational time is considered. The monitor well bottom-hole pressure is recorded at 30 days intervals, consequently, there are 12 observed data, i.e., $N_{d_h} = 12$. As in Emerick and Reynolds [28], we run the reservoir simulator using the true model and add measurement errors to create the vector of observed data $\boldsymbol{d}_{\text{obs}}$; this vector of observed data is identical to the one used by Emerick and Reynolds [28]. The measurement errors are assumed to be independent Gaussian variables with zero mean and variance equal to 1.0 psi^2. Consequently, the measurement error covariance matrix is effectively an identity matrix. Besides the history-matching period, we consider a 390 days forecasting period, with predictions also recorded every 30 days. Therefore, for this case, for every model added to the training set we run the reservoir simulator for a total of 750 days, with data acquired at every 30 days, i.e., $N_d = 25$.

To assess the efficiency and applicability of our proposed method, the results produced with our proposed methodology are compared to the results generated by Li and Reynolds [69]. For the same one-dimensional water-flooding case consider here, Li and Reynolds [69] applied random walk Metropolis-Hastings MCMC, in which the real reservoir simulator predictions was used to evaluate the posterior pdf in order to compute the Metropolis-Hastings acceptance probability to quantify the uncertainty. They generated a Markov chain composed of 23 million states. Their proposal distribution was a Gaussian distribution centered on the current state in the chain and covariance matrix given by $0.005^2 \times C_M$, where C_M represents the prior covariance matrix.

The random walk results of Li and Reynolds [69] are presented in Fig. 4.5. The marginal distributions of gridblock permeability are presented in Fig. 4.5(a). In this figure, and for similar figures shown later in this dissertation, the solid thick black curve represents



(a) (b)

Figure 4.5: Random walk Metropolis-Hastings MCMC results from Li and Reynolds [69]: (a) marginal distributions of gridblock permeability, (b) marginal distributions of water production rate at the production well. Both plots show the corresponding true value (solid thick red curve), the mean of the marginal distribution (solid thick black curve), the percentiles P25 and P75 of the marginal distribution (two blue dashed curves), and the percentiles P5 and P95 of the marginal distribution (two thin black curves).

the mean of the marginal distribution for each gridblock, also the two blue dashed curves represents the 25% (P25) and 75% (P75) percentiles, while the two solid thin black curves represent the 5% (P5) and 95% (P95) percentiles. Finally, the solid thick red curve represents the true permeability field. Fig. 4.5(b) shows the associated marginal distributions for the water production rate at the production well. In this latter figure, the solid thick red curve represents the water production rate that is obtained running the reservoir simulator model with the true permeability field, and the other curves follow the same color code as described for Fig. 4.5(a). It is worthwhile to remember that the water production rate at the production well was not used in the history matching process.

Following our proposed methodology, to find modes of the posterior pdf and build the GMM approximation, we chose to solve a total of $N_e = 200$ minimization problems. The required $N_e = 200$ initial guesses are sampled from the prior pdf using LHS [74]. For this case, an additional $N_a = 600$ models are also sampled from the prior with LHS, to compose an initial training set with $N_t = 800$ training examples.

The minimization loop set up is again based on the values adopted by Rafiee [93].

The minimum distance to update the training set is selected equal to $d_{\min} = 1.0\text{E}-4$, since we normalize the distance using the dimension of the vector $\boldsymbol{m}$, see Eq. 3.25, a numerical values between $1.0\text{E}-4$ and $1.0\text{E}-5$ seems to work well for the problems addressed so far. The trust-region parameters are set equal to $\delta_e^{(1)} = 1.0$, for $e = 1, \ldots, N_e$, $\delta_{\min} = 1.0\text{E}-2$ and $\delta_{\max} = 2.0$, this numerical values seems to work well considering that the minimization process is actually conducted for the dimensionless vector $\hat{\boldsymbol{m}}$, see Eq. 2.11. The convergence criteria are set equal to $\epsilon_{\min} = \epsilon_m = 1.0\text{E}-6$ and $p_{\text{conv}} = 0.95$, i.e., we require that at least 95% of the minimization problems converge. While the minimization loop is conducted, we set $N_{\text{cut}} = 3,200$ and $O_{N,\text{cut}} = 100.0$, which means that when the training set becomes larger than 3,200 training examples, we remove from the training set the training examples which provide normalized objective function values greater than 100.0.

For this one-dimensional problem, we apply our mapping approach to improve the quality of the proxy predictions. To achieve this, we derive a mapping function $\psi(\boldsymbol{m})$ from a simplified one-dimensional analytical solution. The solution which is considered here is for an one-dimensional, single-phase, semi-infinite, homogeneous reservoir. The resulting analytical solution is given by

$$p(x,t) = p_i - \frac{q_w \mu}{k A_r}\left[\left(\frac{4\eta t}{\pi}\right)^{\frac{1}{2}} \exp\left(-\frac{x^2}{4\eta t}\right) - x\text{erfc}\left(\frac{x}{(4\eta t)^{\frac{1}{2}}}\right)\right] . \tag{4.4}$$

In Eq. 4.4, $p(x,t)$ denotes the reservoir pressure at the axial position x, at the time t, p_i denotes the initial reservoir pressure, q_w represents the constant flow-rate, μ denotes the fluid viscosity, k represents the reservoir permeability, A_r denotes the reservoir cross-sectional area, and the hydraulic diffusivity η is given by

$$\eta = \frac{k}{\phi \mu c_t} , \tag{4.5}$$

for any consistent system of units, where ϕ denotes the reservoir porosity and c_t the reservoir total compressibility.

In Eq. 4.4, the second term in brackets, i.e., the term involving the complementary

error function, erfc(·), is dominated by the exponential term. Dropping the complementary error function term, from Eq. 4.4 we get the following dependence relationship between permeability and pressure

$$p(k) \;\propto\; \frac{\sqrt{k}}{k} \exp\left(-\frac{1}{k}\right) . \tag{4.6}$$

We chose the dependence relation of Eq. 4.6 because, for this 1-D water-flooding model, the reservoir parameters are represented by the log-permeability and the observed data by the monitor well pressure. Hence, since the reservoir parameters are the log-permeability $k = \exp(m_j)$. Using this relation in Eq. 4.6 results in the following mapping function

$$\psi(m_j) = \exp\left[-\frac{m_j}{2} - \exp(-m_j)\right] , \quad \text{for} \quad j = 1, 2, \ldots , N_m . \tag{4.7}$$

To make use of the mapping function $\psi(m_j)$ of Eq. 4.7, before we training a LS-SVR proxy model, we apply $\psi(m_j)$ to the entries of the input vector $\boldsymbol{m}$.

To evaluate the derived mapping function and select the value of σ for training the LS-SVR proxy model, as described earlier, we conduct some numerical experiments. The results are presented in Fig. 4.6. As stated earlier, we adopted the value of $\gamma = 800$ in all LS-SVR proxy training procedure. The plot presented in Fig. 4.6 represent the RMSE value for several different values of sigma, see the discussion preceding Eq. 3.15. The blue curve represents the case in which no mapping functions is used, and the red curve represents the results for the case where the mapping function is used. As one can see, the use of the simple mapping function consistently delivers lower values of the RMSE. This is an indication that the mapping function indeed improves the quality of the LS-SVR proxy model, although only marginally. From the behavior presented in Fig. 4.6, we selected the value of $\sigma = 1.40$, which corresponds to $c_\sigma = 0.25$, see Eq. 3.9. To generate the results presented in Fig. 4.6, we use the $N_e = 200$ selected models as the test set to compute the RMSE value, and we use the $N_a = 600$ selected models to create the training set. We opt to split the selected 800 models in this way since each one of these individual sets, i.e., $N_e = 200$ models and $N_a = 600$ models, were independently sampled from the prior pdf, both using LHS.

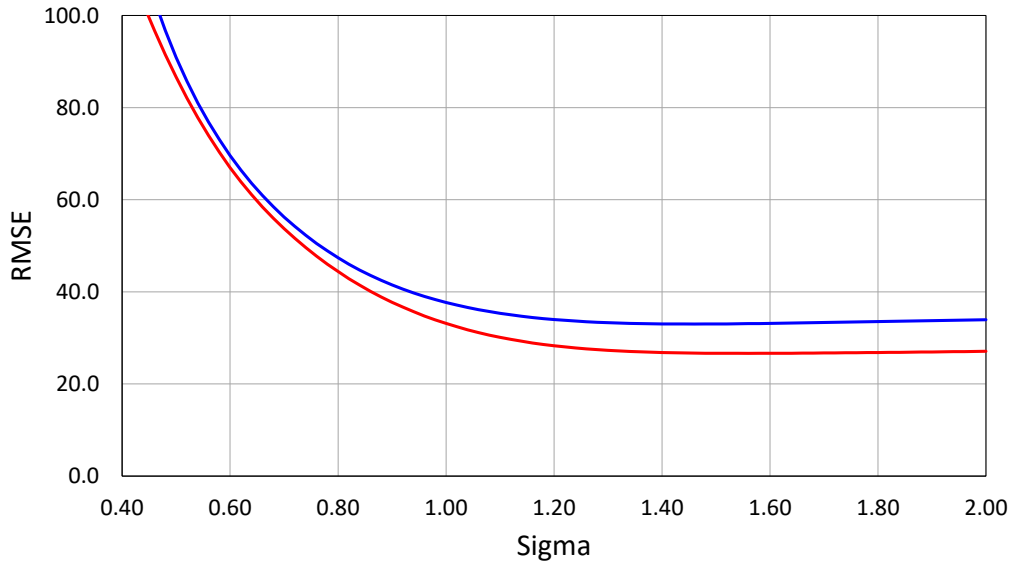


Figure 4.6: LS-SVR proxy behavior when varying the value of σ, for the one-dimensional water-flooding case, comparing the case without using the mapping function (blue curve), with the case using the mapping function (red curve).

In Fig. 4.7 we present the minimization loop results. As one can see, the final values of the normalized objective function obtained after convergence are four orders of magnitude lower than the corresponding values for the prior models. This is an indication that the LS-SVR proxy gradient is indeed a good alternative to the adjoint solution, although Eq. 3.27, which gives the approximate upper bound for the value of the normalized objective function at the minima, which is 3.04 for this case, is satisfied for only nine cases. Nevertheless, the LS-SVR proxy gradient is only a practical approximation and the overall results presented in Fig. 4.7 are more than satisfactory for our applications.

As discussed in Chapter 3, to further improve the computational efficiency, at the end of the minimization process, only the training examples which provide a normalized objective function value less than $O_{N\max} = 80.0$ are selected to form the final training set. Then, we train a final LS-SVR proxy model using the resulting final training set. In order for the minimization loop to converge, a total of 5,219 reservoir simulation runs are required for this case. At the end, only 2,486 training examples give a normalized objective function value less than $O_{N\max} = 80.0$, and these models are selected to train the final LS-SVR proxy model.

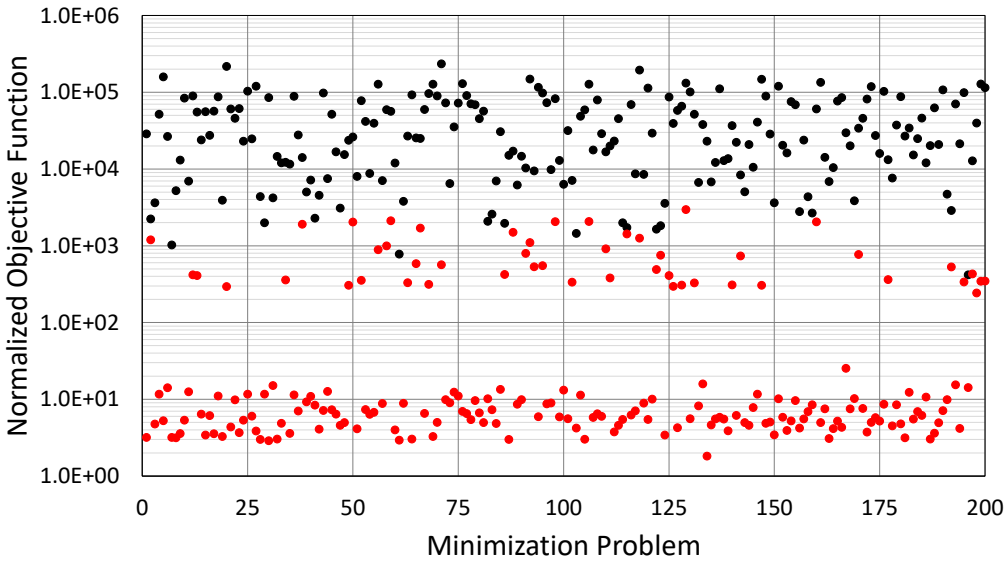
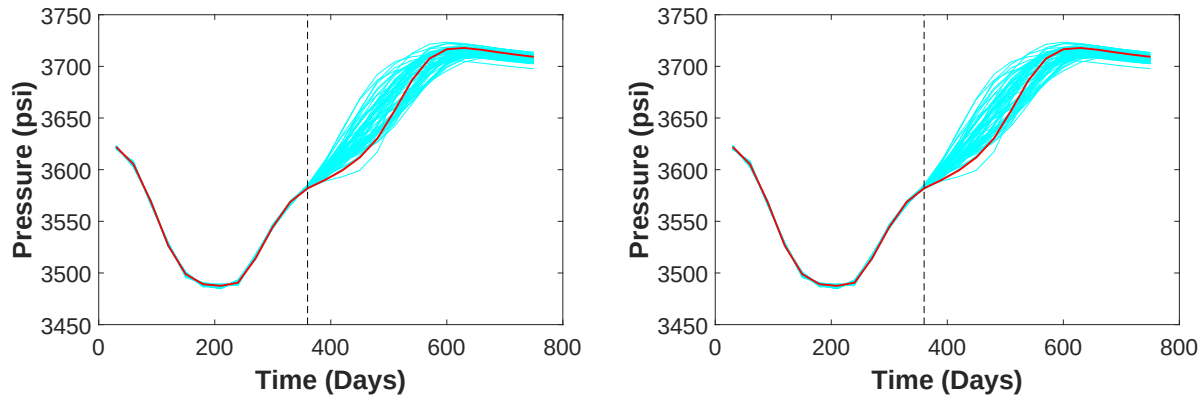


Figure 4.7: Minimization loop results for the one-dimensional water-flooding case: prior models (black dots), converged models (red dots).

Among the 200 minimization problems, 78 have converged to modes that give a normalized objective function value less than 6.0 and are considered to construct the GMM approximation. In Fig. 4.8 we present a comparison for these 78 history-matched models, i.e., these 78 modes of the posterior pdf. The monitor well bottom-hole pressure predictions using the final LS-SVR proxy model are presented in Fig. 4.8(a), while the predictions using the reservoir simulator are presented in Fig. 4.8(b). In each figure, the solid thin cyan curves represent the predictions for the 78 modes found using, respectively, the LS-SVR proxy model, Fig. 4.8(a), and the reservoir simulator, Fig. 4.8(b). In both figures, the solid thick red curve represents the predictions for the true reservoir model using the reservoir simulator. As one can see, there is a good agreement between the final LS-SVR model predictions and the reservoir simulator for all the modes found. It is worth mentioning here that the results of Fig. 4.8(a) are obtained using the mapping function of Eq. 4.7 in the gridblock permeability, not using the mapping function delivers a slightly worse quality of the history matching, as we observed in results no showed in this dissertation.

These 78 modes are clustered into $N_c = 25$ clusters using the k-medoids clustering algorithm [63]. Then, the clusters are used to build a GMM approximation for the posterior pdf. The weights in the GMM approximation are all set equal to $w_c = 1/25 = 0.04$, for



(a) (b)

Figure 4.8: History matched monitor well bottom-hole pressure for the one-dimensional water-flooding case: (a) predictions using the LS-SVR proxy model, (b) predictions using the reservoir simulator for the same converged models as in part (a). In both figures, the true bottom-hole pressure is represented by the solid thick red curve, and the predictions for the history matched models is represented by the cyan curves. The vertical dashed black line divides the history matched and the forecasting periods.

$c = 1, \ldots, N_c$. Using the resulting GMM approximation as the proposal distribution, we run five chains with a total of 100,000 states each. The value of the parameter $\mu_\gamma = 1.0$, see the discussion preceding Eq. 2.65, is selected based on a previous evaluation done by running short chains. This choice of μ_γ results in a constant learning rate of $\gamma_i = 1.04\text{E} - 3$, which delivers an overall acceptance rate of 21.4%. The convergence of the chains based on MPSRF [14] is presented in Fig. 4.9. As one can see, the MPSRF reaches unity after 5,000 states are generated. Thus, we regard the first 5,000 states on each chain as the burn-in period. Hence, states subsequent to state 5,000 in each chain represent samples from the target distribution, i.e., the sought samples from the posterior pdf. As commented earlier, reaching convergence after only 5,000 states are generated, corroborates the good performance of MCMC sampling when using the GMM approximation as the proposal distribution.

The MCMC uncertainty quantification results are presented in Fig. 4.10. The marginal distributions of the gridblock permeability is presented in Fig. 4.10(a), while the associated marginal distributions of the water production rate at the production well is presented in Fig. 4.10(b). These marginal distributions are constructed using the last 80,000 states from

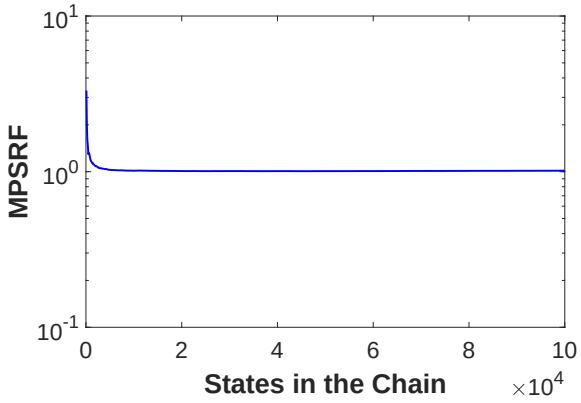


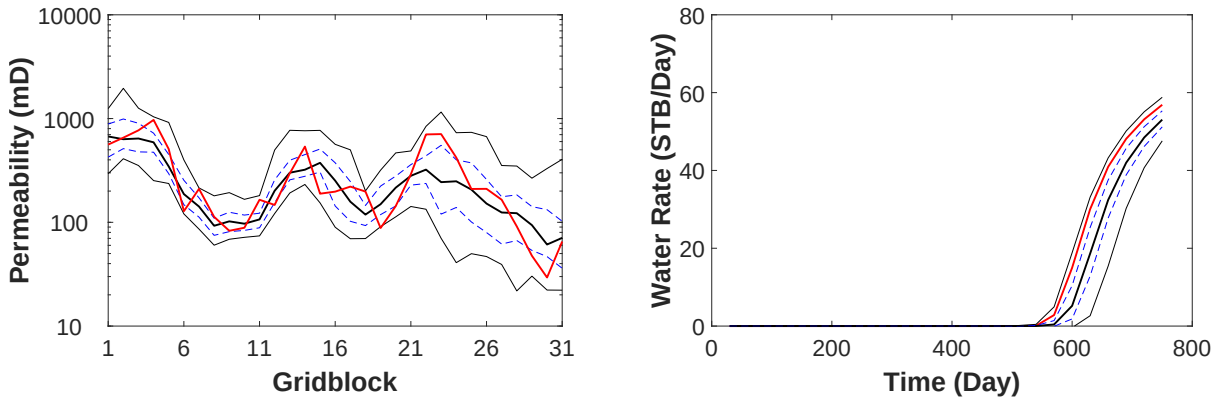
Figure 4.9: MCMC sampling convergence based on the MPSRF for the one-dimensional water-flooding case.

each of the five chains, i.e., in a total of 400,000 states. The comparison between the results presented in Fig. 4.10 with those results from Fig. 4.5 seems to indicate that the methodology proposed here delivers a reasonable uncertainty quantification for this 1-D water-flooding case, with the advantage of employing only a tiny fraction of the reservoir simulations used for the random walk MCMC results of Li and Reynolds [69]. More importantly, our results are generated with a total of 5,219 reservoir simulation runs, while the results of Li and Reynolds [69] using their GMM proposal distribution take over 505,000 reservoir simulation runs.

4.3 Application to a Synthetic Two-Dimensional Reservoir Model.

This synthetic two-dimensional reservoir case was designed by Li [68]. The reservoir simulation model has a $44\times44\times1$ grid, with each gridblock having dimensions of $100\times100\times15$ ft. The initial reservoir pressure is set to 3,000 psi. The reservoir porosity is set to a constant value equal to 0.20 for all gridblocks.

Similar to the previous example, the reservoir model parameters, vector $\boldsymbol{m}$, is given by the natural logarithm of horizontal permeability of each gridblock. The horizontal permeability field is assumed to be isotropic. Therefore, $N_m = 44 \times 44 = 1936$ and $m_j = \ln(k_j)$, for $j = 1, 2, \ldots, N_m$, where k_j represents the permeability of the jth gridblock, and m_j represents the jth entry of the vector $\boldsymbol{m}$. The log-permeability is assumed to follow a



(a) (b)

Figure 4.10: MCMC uncertainty quantification results for the one-dimensional water-flooding case: (a) marginal distributions of gridblock permeability, (b) marginal distributions of water production rate at the production well. The colors and thickness of each curve have the same meaning as in Fig. 4.5. To generate the marginal distributions presented here, all states after the 20,000st state in each of the five chains are used, resulting in a total of 400,000 states.

multivariate Gaussian distribution with mean $\boldsymbol{m}_{\text{pr}} = [5.0, \ldots, 5.0]^T$, so $m_{\text{pr},j} = 5.0$, for $j = 1, 2, \ldots, N_m$. The variance of $\ln(k_j)$ for each gridblock is assumed equal to 1.0. The prior covariance matrix was constructed by Li [68] using an exponential covariance function with major and minor rages of 2,500 ft and 1,100 ft, respectively. The main direction is oriented at 45° to the positive $y-$axis.

The true reservoir model, i.e., the true permeability field generated by Li [68] is presented in Fig. 4.11. The reservoir is operated with 13 wells, 9 production wells and 4 injection wells. The location of each well is also shown in Fig. 4.11. All 4 injection wells are operated at a constant bottom-hole pressure of 4,500 psi. The production wells are operated at a constant bottom-hole pressure of 2,800 psi for the first six months, and then operated at a constant bottom-hole pressure of 2,500 psi for the remaining operational time.

The observed data considered for history-matching include the monthly water injection rate for each injection well and both the monthly water and oil production rates from each production well. A total operational time of 36 months is considered in the history-matching period, with a total of 792 observed data. The history matching period is followed by an additional of 20 months of a forecasting period. Therefore, to include a training exam-

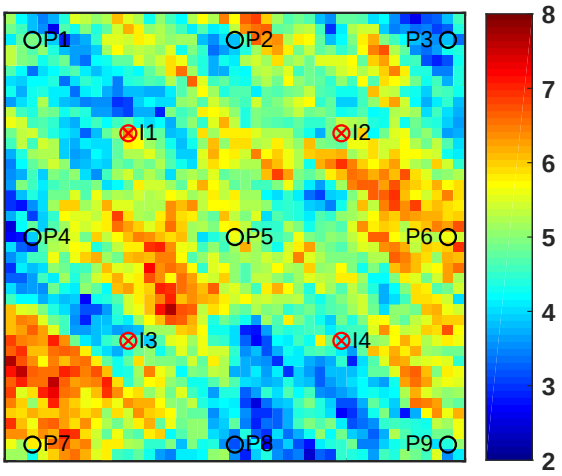


Figure 4.11: The true log-permeability field of Li [68]. The wells locations are also presented: black circles represent production wells, while red circles with a cross represent injection wells.

ple in the training set, we run the reservoir simulator for a total of 56 months. The reservoir simulator was run with the true reservoir model and the results added to measurement errors to produce the vector of observed data, $\boldsymbol{d}_{\text{obs}}$, the same one adopted by Li [68]. The measurement errors are assumed Gaussian, with zero mean and diagonal covariance matrix. To construct the diagonal measurement errors covariance matrix, the standard deviations of the measurement errors are set equal to 5% of the predicted flow-rates obtained by running the true reservoir model. However, a minimum standard deviation of 2.0 STB/day is assumed.

Following the proposed methodology, we solve a total of $N_e = 250$ minimization problems in order to find modes of the posterior pdf and build a GMM approximation. The required $N_e = 250$ initial guesses are sampled from the prior pdf using LHS [74]. An additional $N_a = 500$ models are also sampled from the prior pdf, with LHS, to form an initial training set with $N_t = 750$ training examples. In the minimization loop, we adopt a minimum distance to update the training set of $d_{\min} = 1.0\text{E}-4$, which seems to work well when the distance is normalized using the dimension of the vector $\boldsymbol{m}$, as discussed earlier. The trust-region parameters are set to $\delta_e^{(1)} = 1.0$, for $e = 1, \ldots, N_e$, $\delta_{\min} = 1.0\text{E}-2$ and $\delta_{\max} = 2.0$, which again seems to work well considering that the minimization process is actually conducted for the dimensionless vector $\hat{\boldsymbol{m}}$, see Eq. 2.11.

The convergence criteria are set to $\epsilon_1 = \epsilon_2 = 1.0\text{E}-6$ and $p_{\text{conv}} = 0.95$, so again a minimum of 95% of all minimization problems have to converge. During the minimization loop, we set $N_{\text{cut}} = 4,000$ and $O_{N,\text{cut}} = 80.0$, which means that when the training set becomes larger than 4,000 training examples, then at each iteration of the minimization loop, we remove from the training set the training examples which provide normalized objective function values greater than 80.0, see the discussion preceding Eq. 3.27.

To generate a mapping function for this two-dimensional case, we resort to the analytical solution for an infinite-acting radial flow reservoir geometry, with homogeneous properties and single-phase flow, producing with constant bottom-hole pressure, which is given in the Laplace space by [91]

$$\bar{q}_D(u) = \frac{1}{u} \frac{\sqrt{u}\, \text{K}_1(\sqrt{u})}{\text{K}_0(\sqrt{u})} \,. \tag{4.8}$$

In Eq. 4.8, u denotes the Laplace space variable, $\text{K}_0(\cdot)$ and $\text{K}_1(\cdot)$ represents the modified Bessel function of second kind of orders zero and one [3], respectively, and $\bar{q}_D(u)$ denotes the Laplace transform of the dimensionless wellbore flow-rate, $q_D(t)$, given by

$$q_D(t) = \frac{\mu}{k\, h\, (p_i - p_o)}\, q(t) \,, \tag{4.9}$$

where μ represents the fluid viscosity, k denotes the homogeneous reservoir permeability, h represents the reservoir thickness, p_i denotes the initial reservoir pressure, p_o represents the wellbore constant bottom-hole pressure, and $q(t)$ represents the wellbore flow-rate.

Unfortunately, there is no known analytical inverse transform of Eq. 4.8 to the real domain. Hence, we opted to numerically invert the analytical solution back to the real domain using the Gaver-Stehfest algorithm [114]. The results are presented in Fig. 4.12 for three different production times, 30 days (red curve), 300 days (green curve) and 1,680 days (blue curve). The production time of 30 days corresponds to the first observed datum, while the production time of 1,680 days corresponds to the last prediction datum, therefore, the times selected covers both the history matching and forecasting periods.

Note from Fig. 4.12 that there is no significant difference in the wellbore flow-rate

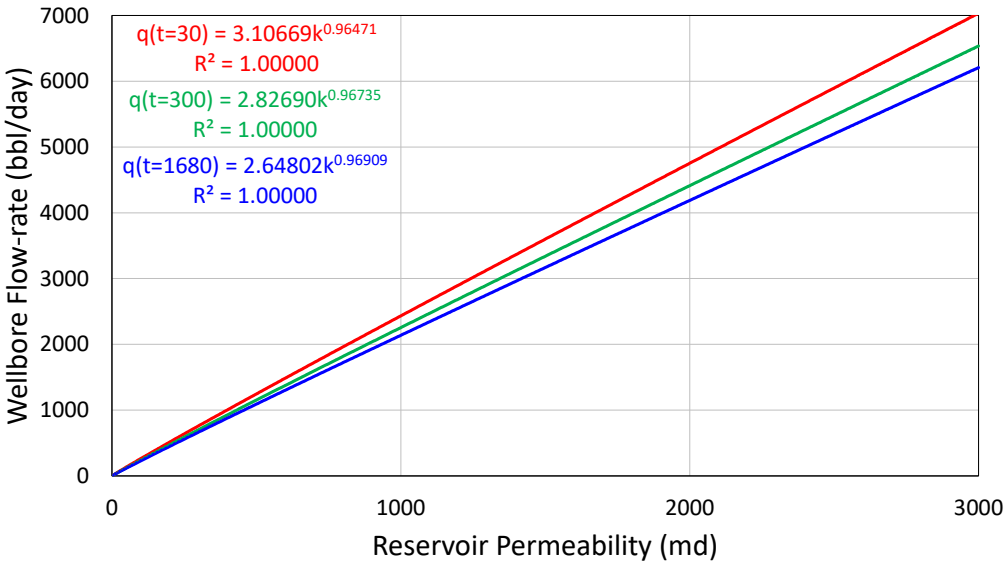


Figure 4.12: Laplace numerical inversion of Eq. 4.8 showing the wellbore flow-rate as a function of the reservoir permeability. The red curve represents a production time of 30 days, the green curve represents a production time of 300 days, and the blue curve represents a production time of 1,680 days. The equations shown in the plot represent curves fitted to the data.

behavior between the three plotted times. To select a mapping function, we opted to average both coefficients and exponents of the fitted curve presented in Fig. 4.12, which results in

$$q(k) = 2.86\, k^{0.967}\,. \tag{4.10}$$

Since for this two-dimensional case the entries of the vector $\boldsymbol{m}$ are given by $m_j = \ln(k_j)$, for $j = 1,\, 2,\, \ldots\,,\, N_m$, from Eq. 4.10, we selected the following mapping function

$$\psi(m_j) = 2.86 \exp\left(0.967\, m_j\right), \quad \text{for} \quad j = 1,\, 2,\, \ldots\,,\, N_m\,. \tag{4.11}$$

As in the previous case, to use the derived mapping, we map the entries of the input vector $\boldsymbol{m}$ using the mapping function defined in Eq. 4.11, before training a LS-SVR proxy model.

As before, to evaluate the derived mapping function and select the value of sigma for training the LS-SVR proxy model, we conduct some numerical experiments. The results are presented in Fig. 4.13. Again, we adopted the value of $\gamma = 800$ for all LS-SVR proxy

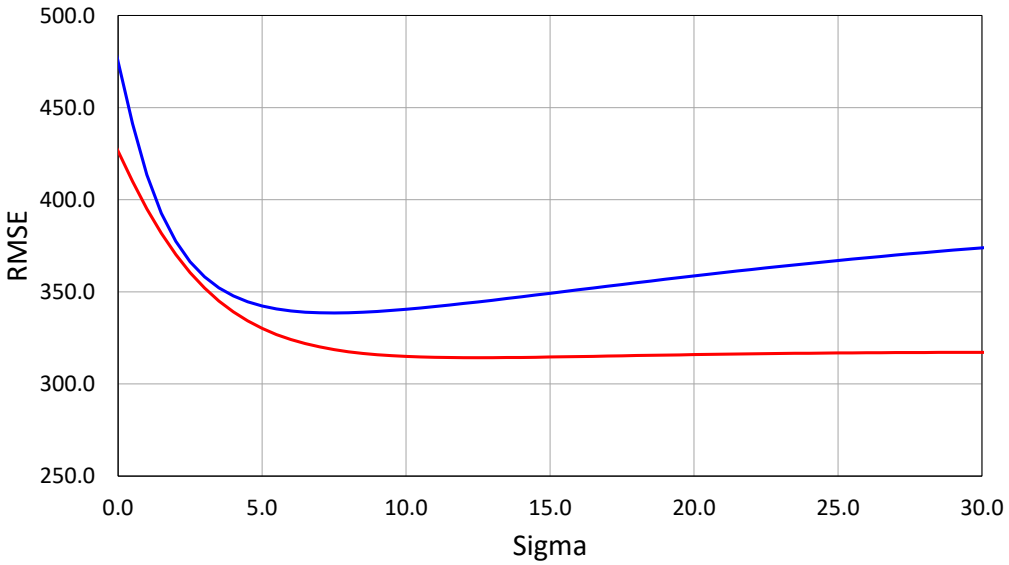


Figure 4.13: LS-SVR proxy behavior when varying the value of σ, for the two-dimensional reservoir case, comparing the case without using the mapping function (blue curve), with the case using the mapping function (red curve).

training procedure. The results presented in Fig. 4.13 represent the RMSE value for several different values of σ, see the discussion preceding Eq. 3.15. The blue curve represents the case in which no mapping functions is used, and the red curve represents the results for the case where the mapping function is used. As before, the use of the simple derived mapping function consistently delivers lower values of the RMSE. This is an indication that the mapping function indeed improves the quality of the LS-SVR proxy model. From the behavior presented in Fig. 4.13, we selected the value of $\sigma = 10.0$, which corresponds to $c_\sigma = 0.227$, see Eq. 3.9. As in the first case, to generate the results presented in Fig. 4.13, we use the $N_e = 250$ selected models as the test set to compute the RMSE value, and we use the $N_a = 500$ extra selected models to create the training set. Again, this split option is based on the fact that the individual sets, i.e., $N_e = 250$ models and $N_a = 500$ models, were independently sampled from the prior pdf.

In Fig. 4.14 we present the minimization loop results, in which the mapping function of Eq. 4.11 is used. As in the previous case, the final values of the normalized objective function obtained after convergence are about four orders of magnitude lower than the corresponding values for the prior models. This corroborates that the LS-SVR proxy gradient is indeed a

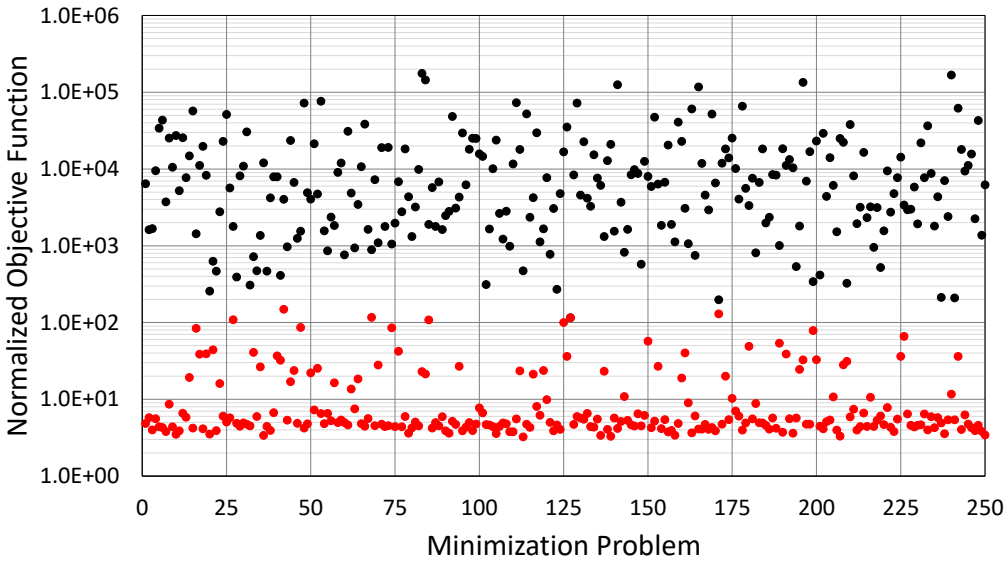


Figure 4.14: Minimization loop results for the two-dimensional reservoir case: prior models (black dots), converged models (red dots).

good alternative to the adjoint solution. However, the Eq. 3.27, which gives the approximate upper bound for the value of the normalized objective function at the minima, provides an upper bound of 1.25, and the minimum value obtained in the minimization loop is 3.24. The performance for the LS-SVR gradient for this case is slightly inferior to the performance observed in the one-dimensional case. Nevertheless, the overall results presented in Fig. 4.14 are more than satisfactory for our applications.

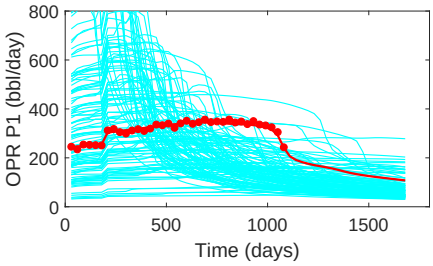
To further improve the computational efficiency, at the end of the minimization process, only the training examples which result in a normalized objective function value less than $O_{N\max} = 20.0$ are selected to form the final training set. Then the final LS-SVR proxy model is trained using the final training set. At the end of the minimization process, a total of 7,001 reservoir simulation runs are required in order to obtain convergence for a minimum of 95% of the 250 minimization problems. The 3309 training examples that gave a normalized objective function value less than $O_{N\max} = 20.0$ are selected to train the final LS-SVR proxy model.

From the 250 minimization problems, 109 converge to modes that give a normalized objective function value less than 4.8 and are considered to construct the GMM approximation. In Figs. 4.15 through 4.19 we present the history matching results for these 109 selected

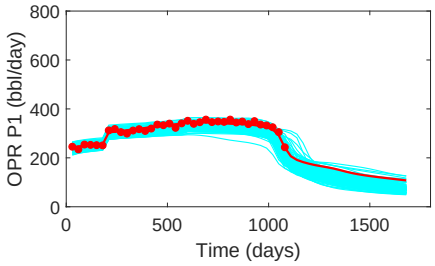
modes. The original prior predictions for the 109 models are compared to the predictions after history matching. As one can see, although the numerical values of the normalized objective function did not reach a value close to 1.25, the overall history matching performance for these 109 selected model are satisfactory. In all figures, the solid thick red curve represents the true value, the red dots represent the observed data, and the cyan curves represents the corresponding models predictions.

The selected 109 modes are clustered into $N_c = 25$ clusters using the k-medoids clustering algorithm [63]. Then, the clusters are used to build a GMM approximation for the posterior pdf. The weights in the GMM approximation again are all set equal to $w_\ell = 1/25 = 0.04$ for $\ell = 1, \ldots, N_c$. Using this GMM approximation as the proposal distribution, we run five chains with a total of 100,000 states each. As before, we first run short chains of length 2,000 states to tune the parameter μ_γ. Running the short chains is a trial and error approach to tune the acceptance rate of the resulting Markov chain. For this case, we selected a parameter $\mu_\gamma = 500.0$, which gives a learning rate of $\gamma_i = 1.33\text{E}-4$, which delivers an overall acceptance rate of approximately 30%. The convergence of the five chains based on the MPSRF method [14] is presented in Fig. 4.20. As one can see, the MPSRF reaches unity after 30,000 states are generated. Therefore, from state 30,000 onward, MPSRF indicates that the chains have converged to the posterior pdf. Consequently, the subsequent states after the state 30,000 are consider samples from the posterior pdf. As before, the GMM approximation delivers a good performance for MCMC sampling procedure, since reaching convergence even after 30,000 states is quite fast.

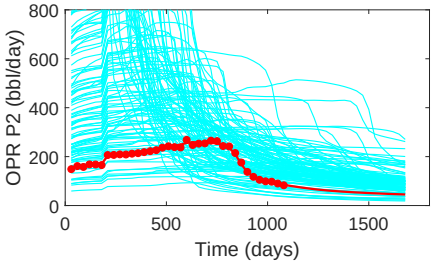
The results when applying our methodology for MCMC uncertainty quantification are compared to the results of Li [68] and Rafiee and Reynolds [95]. For the same two-dimensional case presented here, Li [68] used a reservoir simulator capable of solving the adjoint problem to compute the gradient of the objective function in order to find modes of the posterior pdf. In Li [68] work, the modes found which give a normalized objective function value less than 1.50 were clustered into 25 clusters to build a GMM approximation of the posterior pdf, which was then used as the proposal distribution for MCMC sampling. During the MCMC



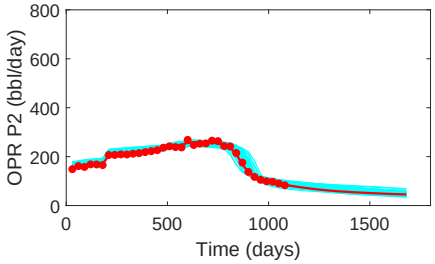
(a) OPR P1: before history matching.



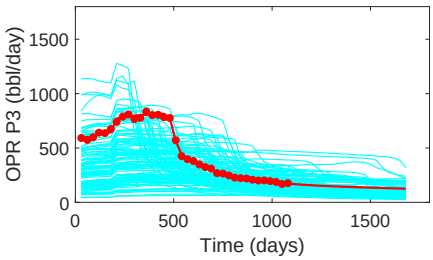
(b) OPR P1: after history matching.



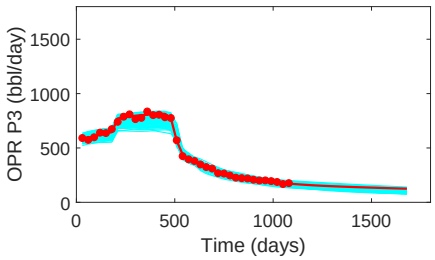
(c) OPR P2: before history matching.



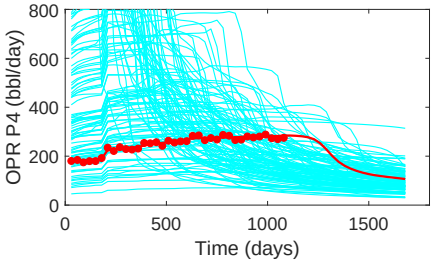
(d) OPR P2: after history matching.



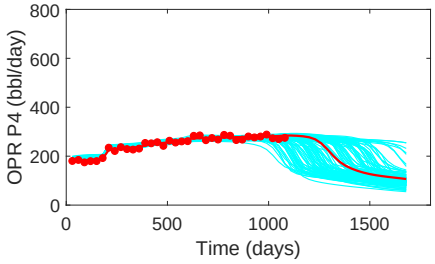
(e) OPR P3: before history matching.



(f) OPR P3: after history matching.

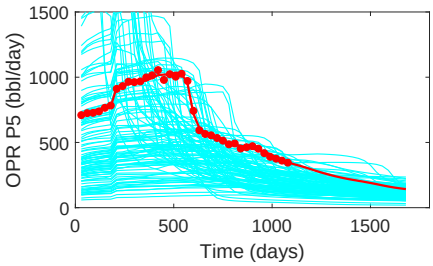


(g) OPR P4: before history matching.

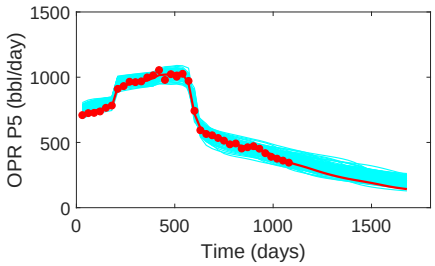


(h) OPR P4: after history matching.

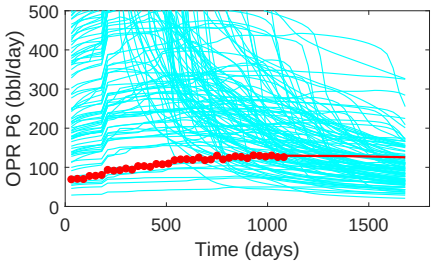
Figure 4.15: History matching results for the two-dimensional reservoir case: oil production rate at producer wells P1 through P4. A comparison between the original prior models predictions (left column) with the predictions after the history matching (right column) is presented. In all figures, we show the corresponding true value (solid thick red curve), the observed data (red dots), and the models predictions (cyan curves).



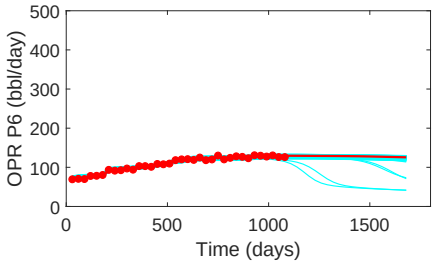
(a) OPR P5: before history matching.



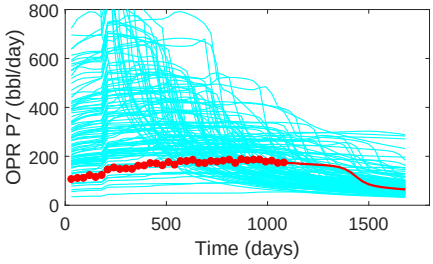
(b) OPR P5: after history matching.



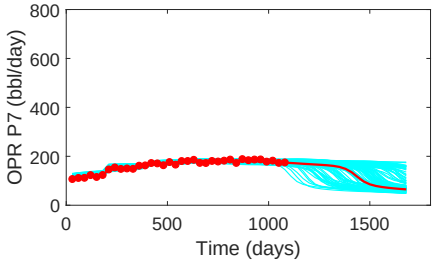
(c) OPR P6: before history matching.



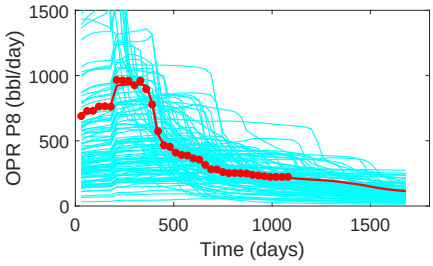
(d) OPR P6: after history matching.



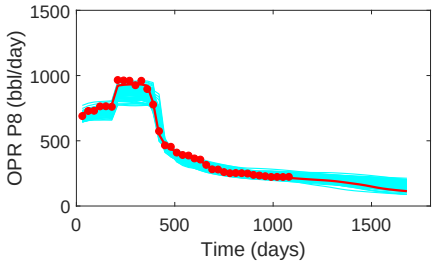
(e) OPR P7: before history matching.



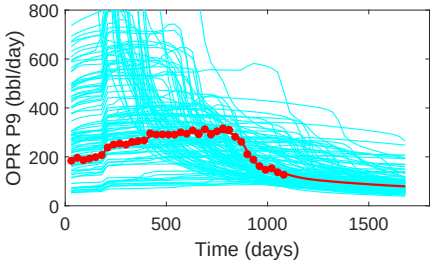
(f) OPR P7: after history matching.



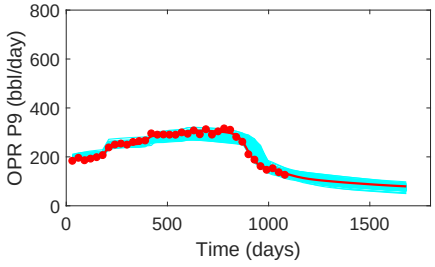
(g) OPR P8: before history matching.



(h) OPR P8: after history matching.

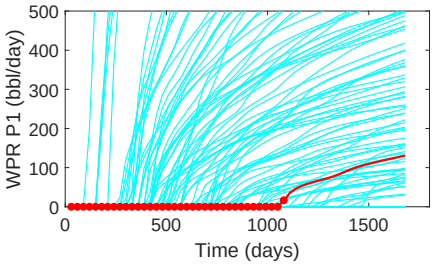


(i) OPR P9: before history matching.

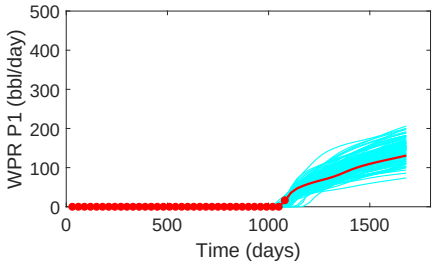


(j) OPR P9: after history matching.

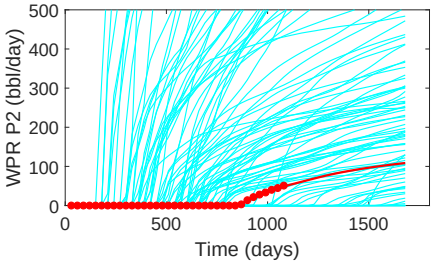
Figure 4.16: Same as in Fig. 4.15 for producer wells P5 through P9.



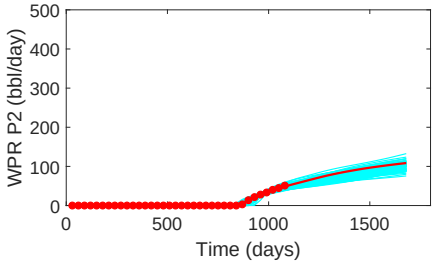
(a) WPR P1: before history matching.



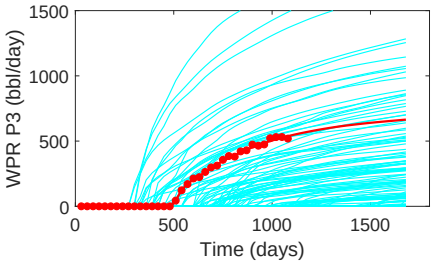
(b) WPR P1: after history matching.



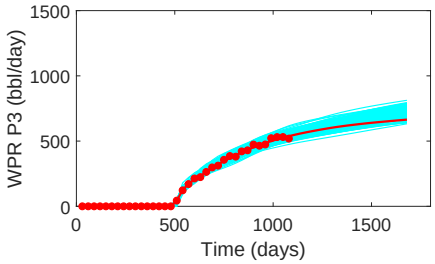
(c) WPR P2: before history matching.



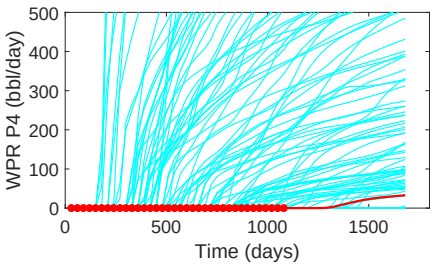
(d) WPR P2: after history matching.



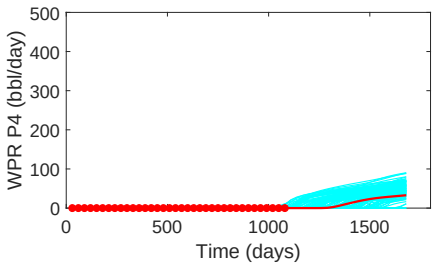
(e) WPR P3: before history matching.



(f) WPR P3: after history matching.

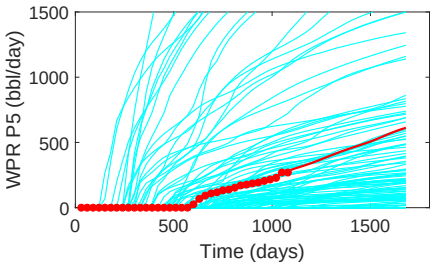


(g) WPR P4: before history matching.

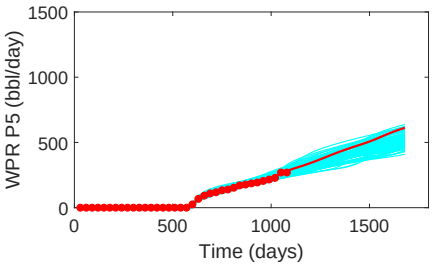


(h) WPR P4: after history matching.

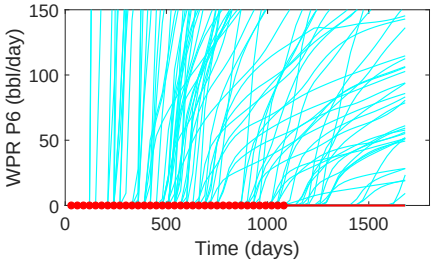
Figure 4.17: History matching results for the two-dimensional reservoir case: water production rate at producer wells P1 through P4. A comparison between the original prior models predictions (left column) with the predictions after the history matching (right column) is presented. In all figures, we show the corresponding true value (solid thick red curve), the observed data (red dots), and the models predictions (cyan curves).



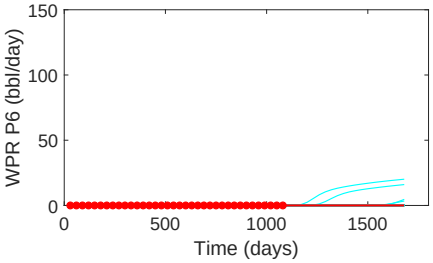
(a) WPR P5: before history matching.



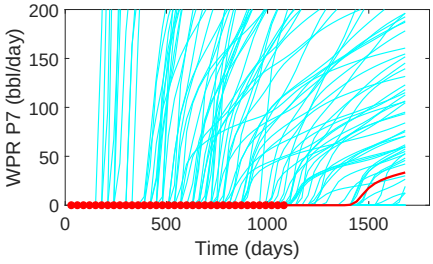
(b) WPR P5: after history matching.



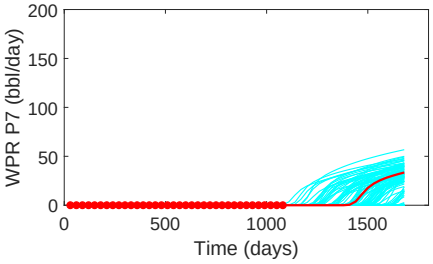
(c) WPR P6: before history matching.



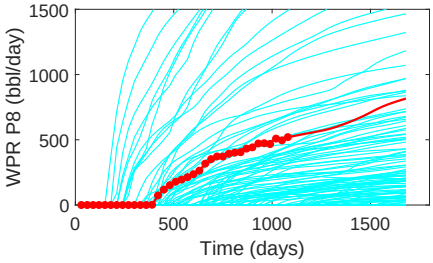
(d) WPR P6: after history matching.



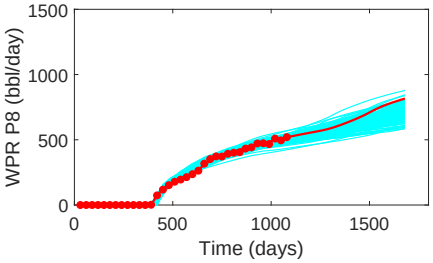
(e) WPR P7: before history matching.



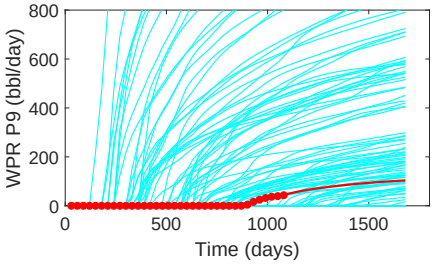
(f) WPR P7: after history matching.



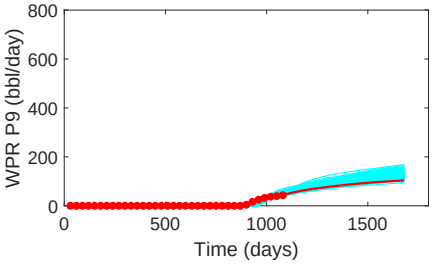
(g) WPR P8: before history matching.



(h) WPR P8: after history matching.

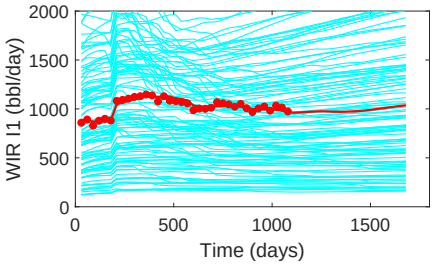


(i) WPR P9: before history matching.

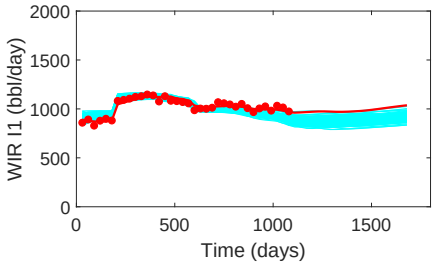


(j) WPR P9: after history matching.

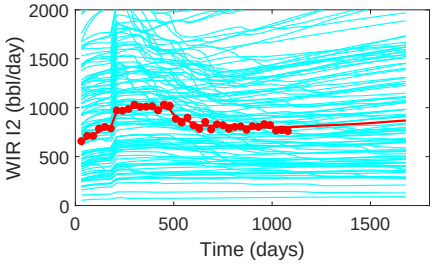
Figure 4.18: Same as in Fig. 4.17 for producer wells P5 through P9.



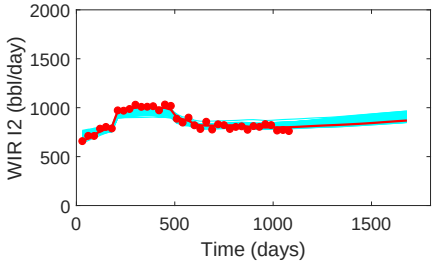
(a) WIR I1: before history matching.



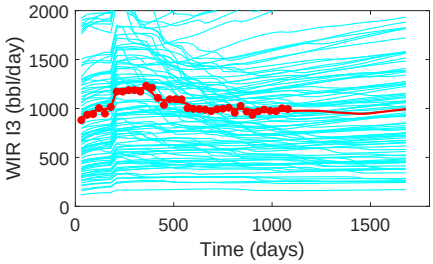
(b) WIR I1: after history matching.



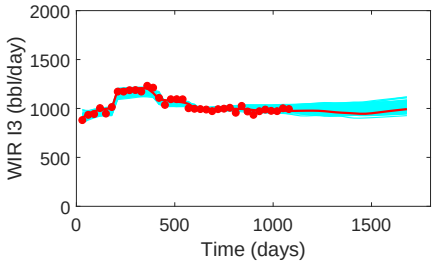
(c) WIR I2: before history matching.



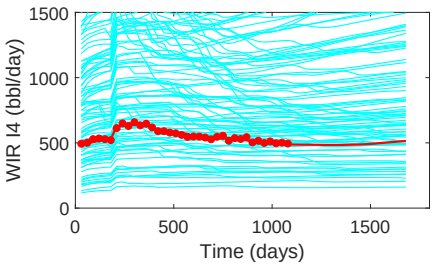
(d) WIR I2: after history matching.



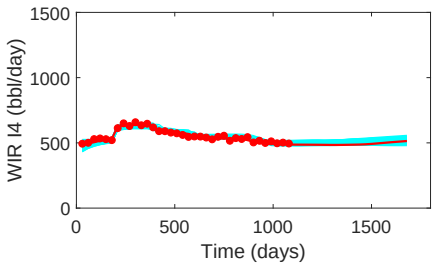
(e) WIR I3: before history matching.



(f) WIR I3: after history matching.



(g) WIR I4: before history matching.



(h) WIR I4: after history matching.

Figure 4.19: History matching results for the two-dimensional reservoir case: water injection rate at injector wells I1 through I4. A comparison between the original prior models predictions (left column) with the predictions after the history matching (right column) is presented. In all figures, we show the corresponding true value (solid thick red curve), the observed data (red dots), and the models predictions (cyan curves).

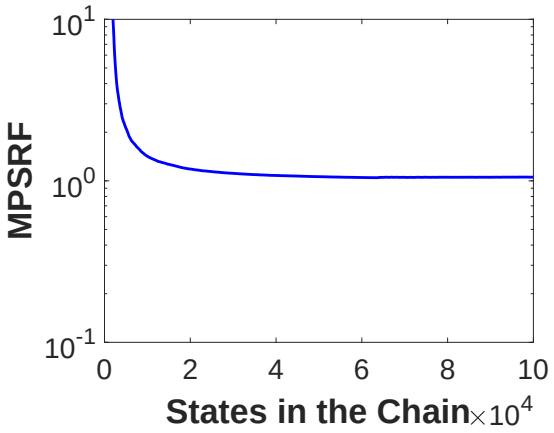
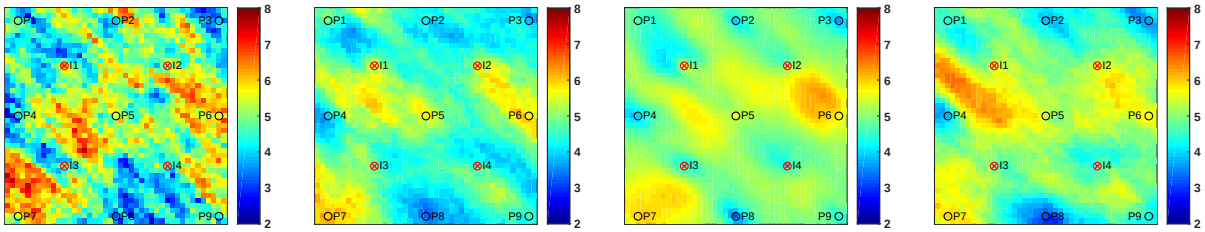


Figure 4.20: MCMC sampling convergence based on the MPSRF for the two-dimensional reservoir case.

procedure, Li [68] used the real reservoir simulator predictions to compute the Metropolis-Hastings acceptance probability, i.e., for each proposed state in the chain, one reservoir simulation run is required. Also for the same two-dimensional case, Rafiee and Reynolds [95] applied the DGN method to approximate the gradient of the objective function in order to find modes of the posterior pdf. In the Rafiee and Reynolds [95] work, the modes found which result in a normalized objective function value less than 7.0 were clustered into 25 clusters to build a GMM approximation of the posterior pdf to use as a proposal distribution. Similar to Li [68], Rafiee and Reynolds [95] then applied the Metropolis-Hastings MCMC algorithm using the reservoir simulator predictions to compute the acceptance probability.

In Fig. 4.21, the posterior mean of the log-permeability field which is obtained when applying our methodology is compared with the true log-permeability field, as well as the posterior mean of Li [68] and Rafiee and Reynolds [95]. As one can see from this figure, the posterior mean of all methods seems to capture some of the features of the true log-permeability field, although it is not possible to draw any firm conclusions on the relative sampling performance of the three methods from the results of Fig. 4.21.

The uncertainty quantification results are presented in Figs. 4.22 through 4.26. The color code shown in all these figures is the same one as described in Fig. 4.5. To reconstruct the marginal distributions presented in Figs. 4.22 through 4.26 we used all states after state 40,000 in the five chains, i.e., a total of 300,000 states.



(a) (b) (c) (d)

Figure 4.21: Posterior log-permeability mean compared to the true log-permeability field: (a) true log-permeability field, (b) this work two-level MCMC, (c) two-level MCMC of Li [68], (d) two-level MCMC of Rafiee and Reynolds [95].

The marginal distribution for oil production rate in all nine producer wells are presented in Figs. 4.22 and 4.23, in these figures the first column presents the results of our two-level MCMC implementation, while the second and third columns present the results of Li [68] and Rafiee and Reynolds [95], respectively. As one can see, the results of our methodology using the LS-SVR proxy model are reasonably consistent with the results of Li [68], who used the reservoir simulator to evaluate the Metropolis-Hastings acceptance probability, except for production well P6, where our results gives far more uncertainty in the oil rate than does the Li [68] method. Note that, other than well P6, Li [68] obtain only a slightly smaller deviation between observed data and corresponding data predicted from the history-matched models, even though Li [68] use a reservoir simulator with adjoint capability for the gradient-based history-matching procedure.

The marginal distribution for water production rate in all nine producer wells are presented in Figs. 4.24 and 4.25. As before, the first, second and third columns, respectively, represent the results of this work, Li [68] and Rafiee and Reynolds [95]. Comparing the marginal distribution for water production rate, one can see that our results are fairly consistent with the results of Li [68].

Finally, Fig. 4.26 shows the marginal distribution for the water injection rate at all four injector wells. The first, second and third columns, respectively, represent the results of this work, Li [68] and Rafiee and Reynolds [95]. Although our procedure produces a wider band of predicted water injection rates for wells I2 and I4 than is obtained by Li [68], the

overall difference between the two sets of results is not practically large.

Until now, we have not commented on the Rafiee and Reynolds [95] results for this two-dimensional case. However, we can see that, overall, the Rafiee and Reynolds [95] results tend to produce a slightly wider uncertainty band than our method. It seems that when wide uncertainty occurs, it appears primarily due to the slightly worse history-match obtained by Rafiee and Reynolds [95]. The advantage that the Rafiee and Reynolds [95] method has over the procedure of Li [69] is that no adjoint gradient is required. Like the work of Rafiee and Reynolds [95], the method presented here does not require an adjoint gradient. Furthermore, and most importantly, unlike the other two methods, our uncertainty quantification workflow does not require any reservoir simulation runs when constructing the Markov chain. It is worthwhile to stress here that our results are generating using 7,001 reservoir simulation runs, while Li [68] and Rafiee and Reynolds [95] results required over 510,000 reservoir simulation runs.

4.4 Application to a Three-Dimensional Model: the PUNQ-S3 Case.

In this section, we apply our developed methodology to a slight modification of the well-known PUNQ-S3 model [36]. For this application, we modify the original well operational conditions. Furthermore, we consider only the horizontal and vertical reservoir natural logarithm permeabilities of gridblocks as the history matching parameters. The PUNQ-S3 model model was developed based on a real field example which was made available by Elf Exploratory & Production. The original reservoir model, geological description and overall documentation can be found at the website

https://www.imperial.ac.uk/earth-science/research/research-groups/perm/standard-models.

The reservoir model is composed of five layers which are arranged in $19 \times 28 \times 5$ gridblocks. The reservoir model has a total of 1,761 active gridblocks. The reservoir geometry is modeled using a corner-point mesh. The average areal gridblock dimensions are $590.55\text{ft} \times 590.55\text{ft}$, which is equivalent to $180\text{m} \times 180\text{m}$. The vertical thickness of gridblocks corresponds to the local vertical thickness of the layer in which the gridblock is locates, which

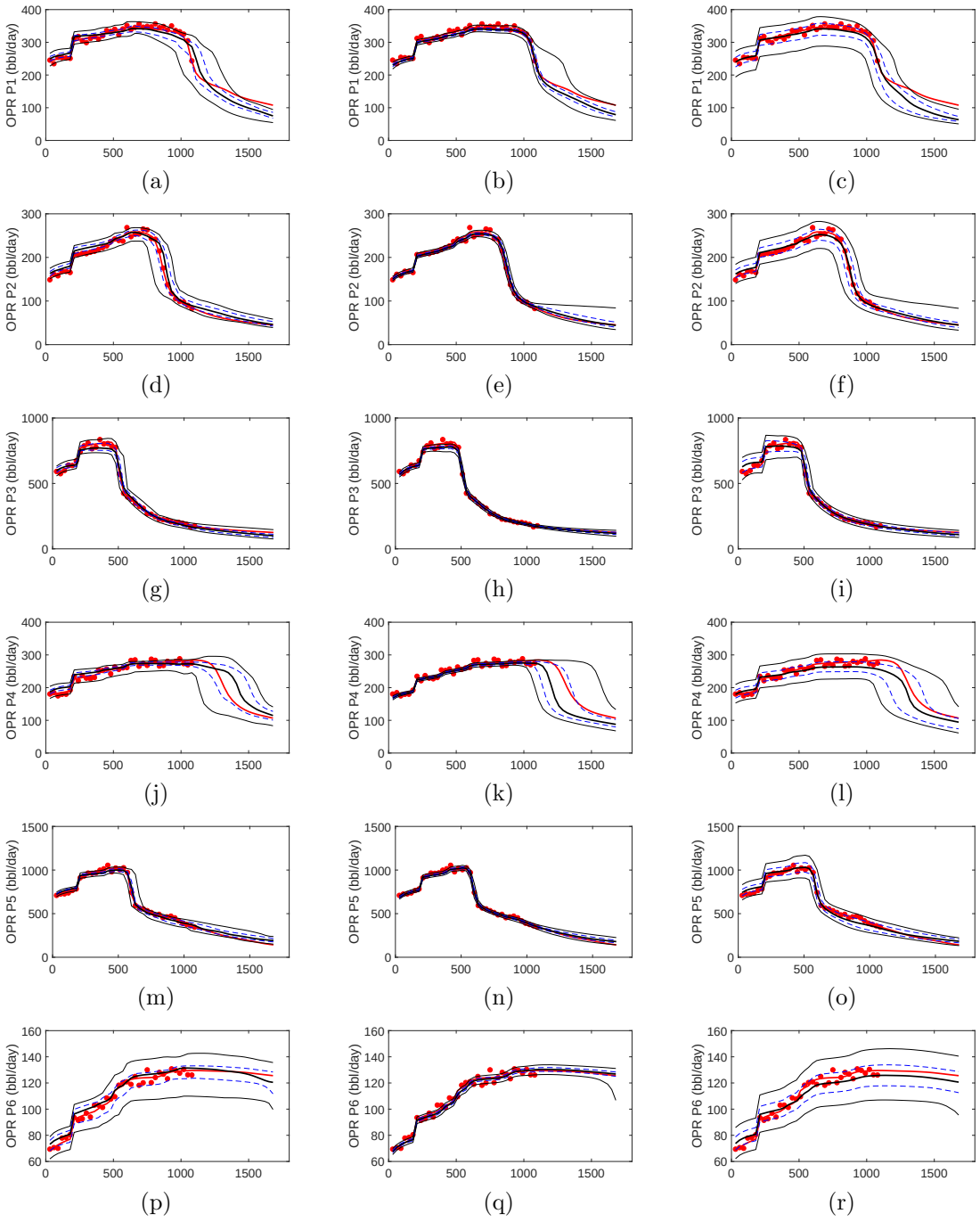


Figure 4.22: MCMC uncertainty quantification results for the two-dimensional case, comparing the marginal distribution for oil production rate of producer wells P1 through P6. The first, second and third columns present the results of this work, Li [68] and Rafiee and Reynolds [95], respectively. In all figures, we show the corresponding true value (solid thick red curve), the observed data (red dots), the mean of the marginal distribution (solid thick black curve), the percentiles P25 and P75 of the marginal distribution (two blue dashed curves), and the percentiles P5 and P95 of the marginal distribution (two thin black curves). To generate the marginal distributions presented here, all states after state 40,000 in the five chains are used, which gives a total of 300,000 states.

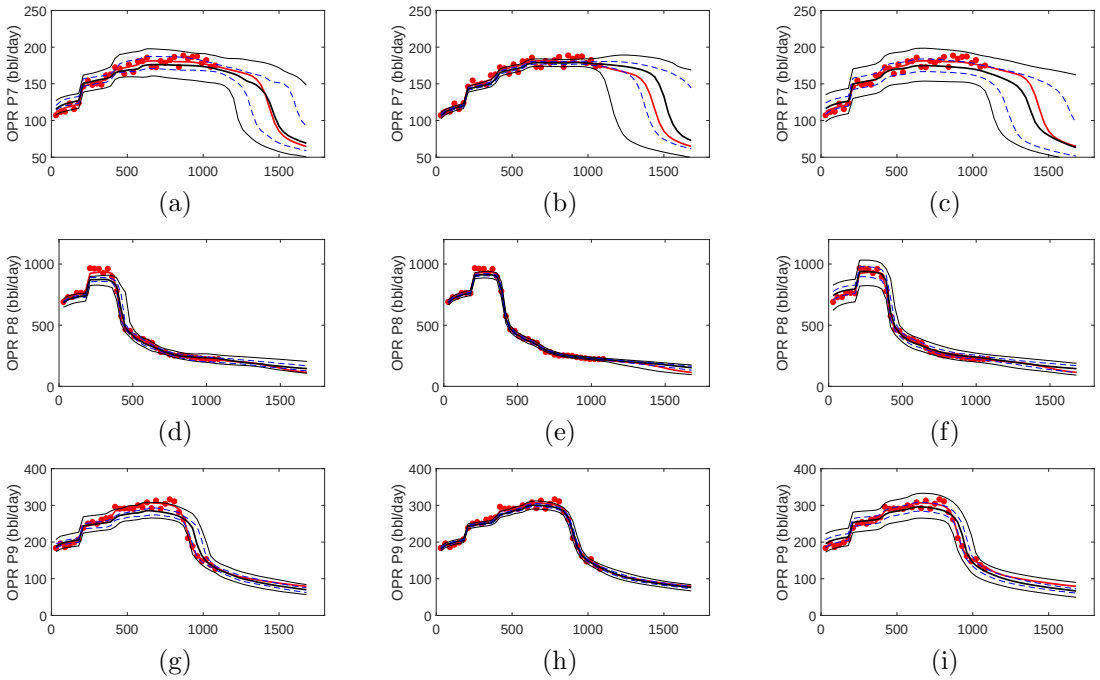


Figure 4.23: Same as Fig. 4.22 for producer wells P7 through P9. The colors and thickness of each curve have the same meaning as in Fig. 4.22.

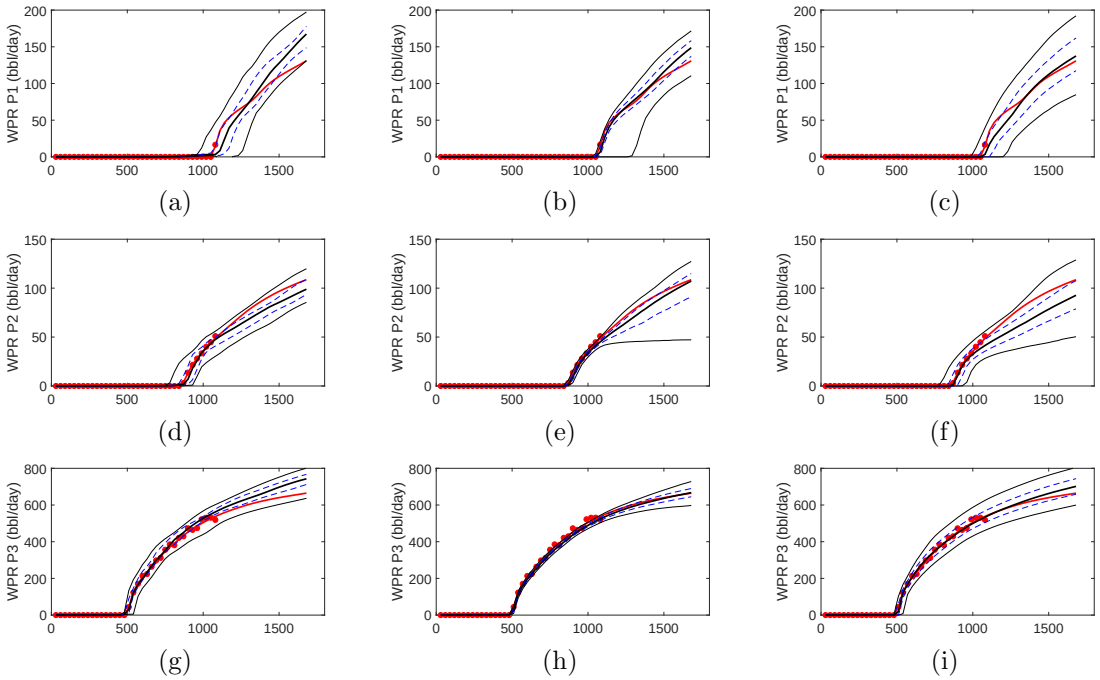


Figure 4.24: MCMC uncertainty quantification results for the two-dimensional case, comparing the marginal distribution for water production rate of producer wells P1 through P3. The first, second and third columns present the results of this work, Li [68] and Rafiee and Reynolds [95], respectively. The colors and thickness of each curve have the same meaning as in Fig. 4.22. To generate the marginal distributions presented here, all states after state 40,000 in the five chains are used, which gives a total of 300,000 states.

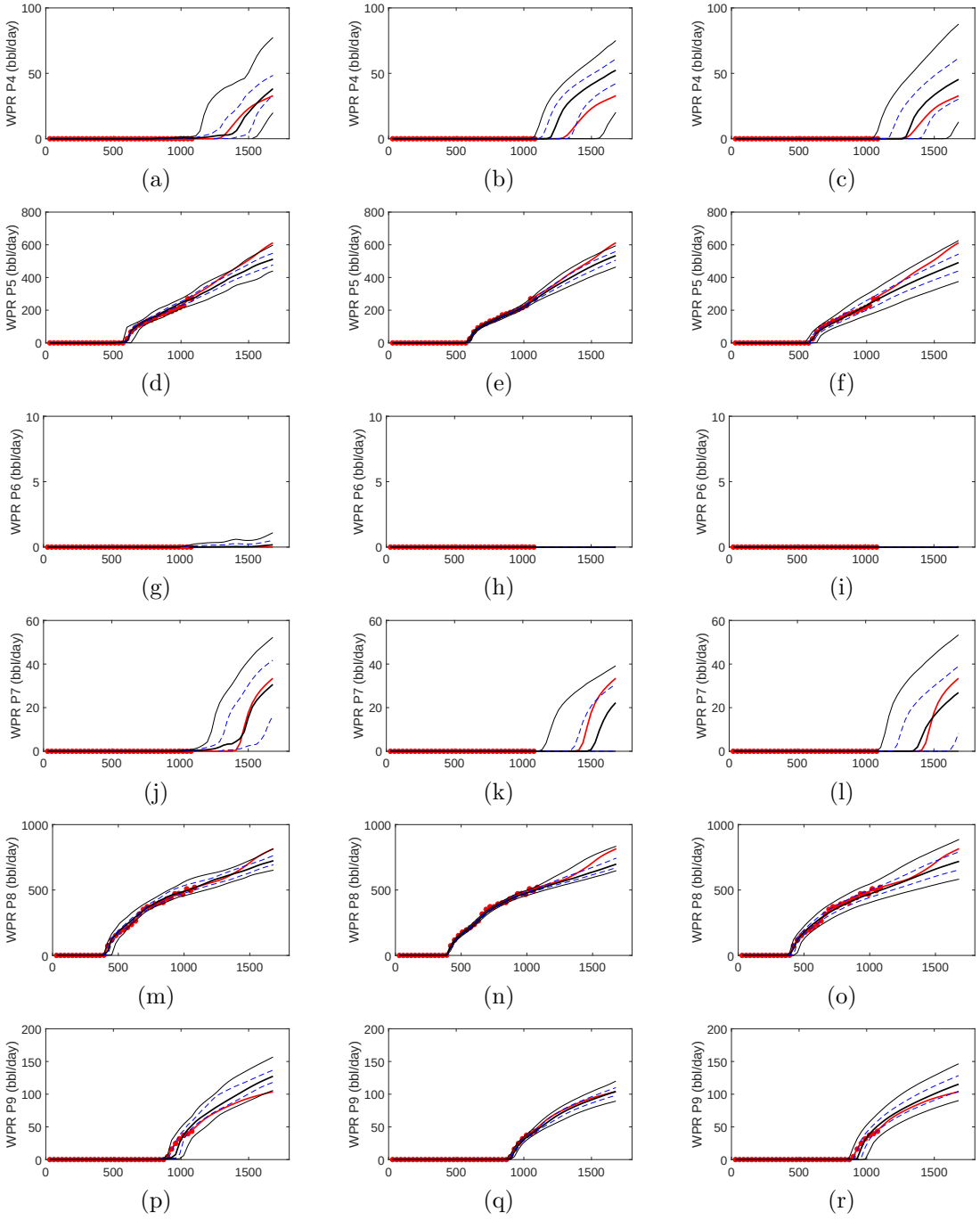


Figure 4.25: MCMC uncertainty quantification results for the two-dimensional case, comparing the marginal distribution for water production rate of producer wells P4 through P9. The first, second and third columns present the results of this work, Li [68] and Rafiee and Reynolds [95], respectively. The colors and thickness of each curve have the same meaning as in Fig. 4.22. To generate the marginal distributions presented here, all states after state 40,000 in the five chains are used, which gives a total of 300,000 states.

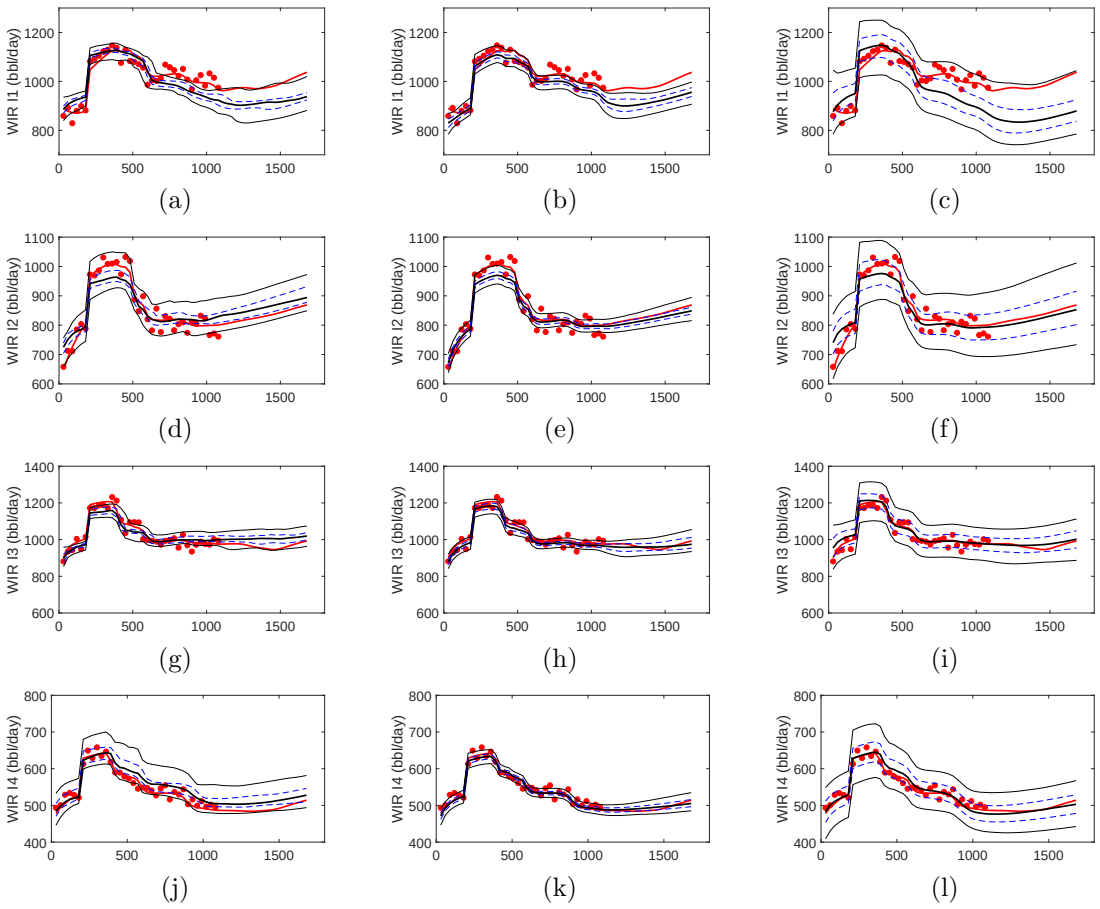


Figure 4.26: MCMC uncertainty quantification results for the two-dimensional case, comparing the marginal distribution for water injection rate of injector wells I1 through I4. The first, second and third columns present the results of this work, Li [68] and Rafiee and Reynolds [95], respectively. The colors and thickness of each curve have the same meaning as in Fig. 4.22. To generate the marginal distributions presented here, all states after state 40,000 in the five chains are used, which gives a total of 300,000 states.

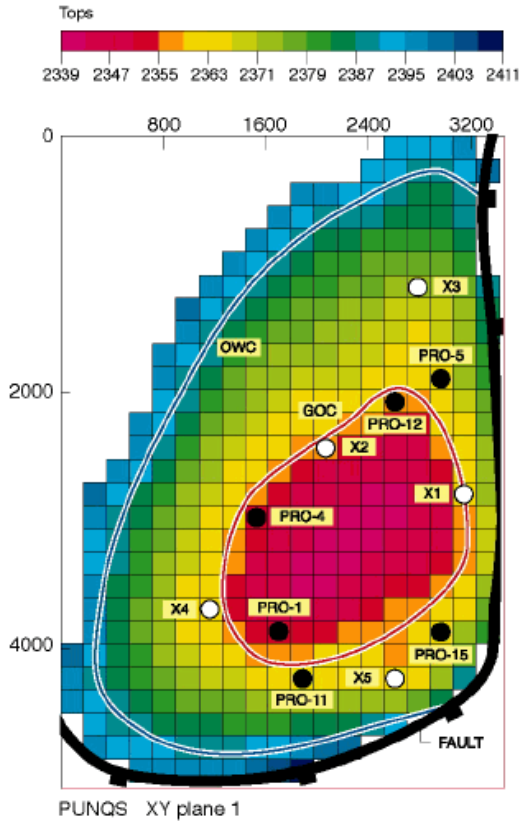


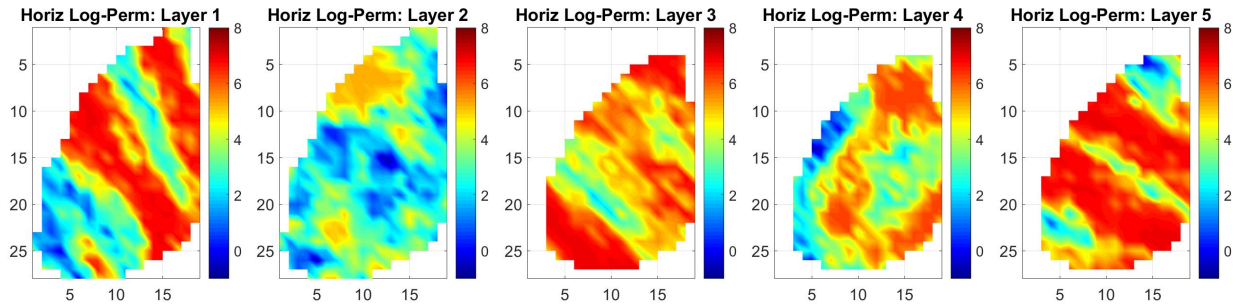
Figure 4.27: Top structure of the PUNQ-S3 model, also presenting the well locations.

varies between 4 ft and 30 ft. In Fig. 4.27, which can be obtained in the website cited above, we present the top structure of the PUNQ-S3 model. Fig. 4.27 also shows the well locations. There are a total of six production wells which have been denoted as PRO-01, PRO-04, PRO-05, PRO-11, PRO-12, and PRO-15, the five wells labeled X1, X2, X3, X4 and X5 are infill wells from the original study which are not considered in this research. In Table 4.1, we present the well completion information, i.e., the gridblocks where the wells are open for flow from the reservoir. Different from the original PUNQ-S3 case, in this research, we operate producers PRO-01, PRO-04, PRO-05, PRO-11, and PRO-12 at a constant oil flow rate of 628.98 STB/day, which is equivalent to 100 m^3/day. Producer PRO-15 is operated at a constant oil flow rate of 503.19 STB/day, which is equivalent to 80 sm^3/day.

As noted earlier, for the PUNQ-S3 case we consider history matching the natural logarithm of both horizontal and vertical reservoir permeability for each active gridblock. Consequently, the vector $\boldsymbol{m}$ is composed of the horizontal and vertical log-permeabilities,

Well	gridblock Completion		
	$x-$dir	$y-$dir	$z-$dir
PRO-01	10	22	4 and 5
PRO-04	9	17	4 and 5
PRO-05	17	11	3 and 4
PRO-11	11	24	3 and 4
PRO-12	15	12	4 and 5
PRO-15	17	22	4

Table 4.1: Producer wells gridblock completion for the PUNQ-S3 case.

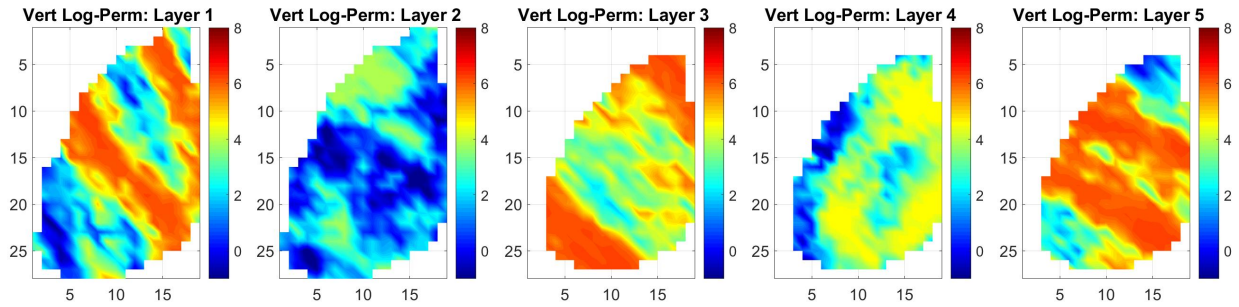


(a) (b) (c) (d) (e)

Figure 4.28: True horizontal log-permeability field for the PUNQ-S3 case: (a) first layer, (b) second layer, (c) third layer, (d) fourth layer, and (e) fifth layer.

and $N_m = 2 \times 1,761 = 3,522$. Furthermore, for $j = 1, 2, \ldots, 1,761$, $m_j = \ln(k_j)$, where m_j represents the jth entry of the vector $\boldsymbol{m}$, and k_j represents the horizontal permeability of the jth gridblock. Similarly, for $j = 1,762, 1,763, \ldots, N_m$, $m_j = \ln(k_{z,j})$, where $k_{z,j}$ denotes the vertical permeability of the jth gridblock. Both the horizontal and vertical true permeability fields adopted here are the same as those developed by Gao et al. [37, 38] and are presented in Figs. 4.28 and 4.29. A detailed description of how the true permeability fields were generated can be found in Gao et al. [37, 38]. It is worth mentioning that Gao et al. [37, 38] constructed a pdf different from the prior pdf in order to generate the true permeability field, hence, the true permeability field is not a sample from the prior pdf.

The prior pdf is assumed to be a multivariate Gaussian distribution. To construct the prior covariance matrix each reservoir layer is assumed independent, i.e., it is assumed that there is no vertical correlation between layers for both the horizontal and vertical log-permeability fields. The individual covariance for both horizontal and vertical log-



(a) (b) (c) (d) (e)

Figure 4.29: True vertical log-permeability field for the PUNQ-S3 case: (a) first layer, (b) second layer, (c) third layer, (d) fourth layer, and (e) fifth layer.

permeability is computed using a Gaussian covariance function. Since no vertical correlation is considered, the Gaussian covariance function, within each layer, depends only on the horizontal distance between gridblocks. For a given reservoir layer, representing the coordinates of its jth gridblock on a horizontal $xy-$plane as the vector $[x_j,\, y_j]^T$, and representing the coordinates of its ℓth gridblock as $[x_\ell,\, y_\ell]^T$, the corresponding covariance between gridblocks j and ℓ is compute by

$$C_{j,\ell}^{\ln(k)}(h_{j,\ell}) = C_{j,\ell}^{\ln(k_v)}(h_{j,\ell}) = \sigma_{xy}^2 \, \exp\left(-3\, h_{j,\ell}^2\right), \tag{4.12}$$

with $h_{j,\ell}$ given by

$$h_{j,\ell} = \sqrt{\left[\frac{x_j^{'} - x_\ell^{'}}{r_x}\right]^2 + \left[\frac{y_j^{'} - y_\ell^{'}}{r_y}\right]^2}, \tag{4.13}$$

where

$$x_a^{'} = \cos(\alpha_{xy})\, x_a \; + \; \sin(\alpha_{xy})\, y_a\,, \qquad \text{for} \quad a = j \;\text{ and }\; \ell\,, \tag{4.14}$$

and

$$y_a^{'} = \cos(\alpha_{xy})\, y_a \; - \; \sin(\alpha_{xy})\, x_a\,, \qquad \text{for} \quad a = j \;\text{ and }\; \ell\,. \tag{4.15}$$

In Eqs. 4.12 through 4.15, $C_{j,\ell}^{\ln(k)}$ denotes the horizontal log-permeability covariance between gridblocks j and ℓ, similarly, $C_{j,\ell}^{\ln(k_v)}$ denotes the vertical log-permeability covariance between gridblocks j and ℓ, and $h_{j,\ell}$ represents the horizontal distance between gridblocks j and ℓ

Parameter	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5
σ_{xy}	2.49	0.93	2.49	1.35	2.49
α_{xy}	-40	0	-40	40	-40
r_x (m)	800	2,520	1,000	2,750	2,000
r_y (m)	5,040	2,520	5,040	4,125	5,040

Table 4.2: Geostatistical data to compute horizontal and vertical log-permeability covariance for the PUNQ-S3 case.

in an equivalent isotropic log-permeability Gaussian field. Moreover, σ_{xy} represents the variance for both the gridblock horizontal and vertical log-permeability, the vectors $[x_j^{'},\, y_j^{'}]^T$ and $[x_\ell^{'},\, y_\ell^{'}]^T$ denote, respectively, the rotation of the vectors $[x_j,\, y_j]^T$ and $[x_\ell,\, y_\ell]^T$, which are aligned with the log-permeability principal directions, i.e., the coordinate vectors in the equivalent isotropic log-permeability Gaussian field, r_x and r_y denote, respectively, the minimum and maximum correlation ranges, and α_{xy} represents the angle between the positive $y-$direction and the direction with maximum correlation range. The values used to compute $C_{j,\ell}^{\ln(k)}$ and $C_{j,\ell}^{\ln(k_v)}$ are the same given in Gao et al. [37, 38] and are presented in Table 4.2.

The cross-covariance between horizontal and vertical log-permeability, within each layer, is computed using [23, 96]

$$C_{j,\ell}^{\ln(k),\ln(k_v)} = \rho_{k,k_v}\, C_{j,\ell}^{\ln(k)}\ . \tag{4.16}$$

In Eq. 4.16, $C_{j,\ell}^{\ln(k),\ln(k_v)}$ denotes the cross-covariance between horizontal and vertical log-permeability for gridblocks j and ℓ, and ρ_{k,k_v} represents the correlation coefficient between horizontal and vertical log-permeability. In this research, we follow Gao [37] and adopt the value $\rho_{k,k_v} = 0.83$. With the covariance and cross-covariance data for each layer, one can assemble the prior covariance matrix. For the prior mean, we assume a constant value within each layer, where the means within each layer are presented in Table 4.3 and are the same as in Gao [37].

As pointed out by Gao et al. [37, 38], the samples generated using the prior pdf result in some unrealistically low or high values of log-permeabilities. To circumvent the issue, Gao

Prior Mean	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5
$[\ln(k)]_{\text{mean}}$	4.54	3.91	4.54	4.37	4.54
$[\ln(k_z)]_{\text{mean}}$	3.55	2.76	3.55	3.04	3.55

Table 4.3: Adopted constant prior mean per each reservoir layer for the PUNQ-S3 case.

Limit Value	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5
$\ln(k)_{\text{maxup}}$	6.908	5.296	6.908	6.217	6.908
$\ln(k)_{\text{limup}}$	6.217	4.605	6.217	5.756	6.217
$\ln(k)_{\text{limlow}}$	0.000	0.000	0.000	0.000	0.000
$\ln(k)_{\text{maxlow}}$	-2.303	-2.303	-2.303	-2.303	-2.303
$\ln(k_v)_{\text{maxup}}$	6.217	3.914	6.217	4.605	6.217
$\ln(k_v)_{\text{limup}}$	5.296	3.224	5.296	3.914	5.296
$\ln(k_v)_{\text{limlow}}$	0.000	0.000	0.000	0.000	0.000
$\ln(k_v)_{\text{maxlow}}$	-2.303	-2.303	-2.303	-2.303	-2.303

Table 4.4: Low and high trim data limit values, see Eqs. 4.17 and 4.18, for the PUNQ-S3 case.

et al. [37, 38] proposed to trim the generated models using the following procedure

$$\xi_{\text{new}} = \xi_{\text{maxup}} \; + \; \big(\xi_{\text{limup}} - \xi_{\text{maxup}}\big) \exp\left[-\frac{\xi - \xi_{\text{limup}}}{\xi_{\text{maxup}} - \xi_{\text{limup}}}\right] , \qquad \text{for} \quad \xi > \xi_{\text{limup}} , \tag{4.17}$$

and

$$\xi_{\text{new}} = \xi_{\text{maxlow}} \; + \; \big(\xi_{\text{limlow}} - \xi_{\text{maxlow}}\big) \exp\left[-\frac{\xi - \xi_{\text{limlow}}}{\xi_{\text{maxlow}} - \xi_{\text{limlow}}}\right] , \qquad \text{for} \quad \xi < \xi_{\text{limlow}} . \tag{4.18}$$

In Eqs. 4.17 and 4.18, ξ stands for either $\ln(k)$ or $\ln(k_v)$, and the corresponding low and high trim data limit values, i.e., ξ_{maxlow}, ξ_{limlow}, ξ_{maxup} and ξ_{limup}, are given in Table 4.4 [37].

For this PUNQ-S3 case, the data considered for history matching are bottom-hole pressures, the gas production rates and the water production rates for each one of the six producers, where these data are acquired every 30 days. A total of $3 \times 96 = 288$ observed data per well is considered, which is equivalent to a $96 \times 30 = 2,880$ days, or almost a 8 year history matching period. Consequently, there are $N_{d_h} = 3 \times 96 \times 6 = 1,728$ observed production data. The history matching period is followed by $48 \times 30 = 1,440$ days, or almost 4 years, of forecast period. Therefore, $N_d = 3 \times (96 + 48) \times 6 = 2,592$. As a consequence,

to add a point to the training set, we run the reservoir simulator for $(96+48)\times 30 = 4,320$ days, or almost 12 years. The standard deviation of measurement errors for the bottom-hole pressure is assumed equal to 10 bar for all six producers, which corresponds to approximately 4 to 5% of the actual observed data. Meanwhile, the standard deviation of measurement errors for gas production rate is assumed equal to 50 m^3/day for producers PRO-01 and PRO-04, and assumed equal to 200 m^3/day for producers PRO-05, PRO-11, PRO-12, and PRO-15, which corresponds to approximately 3% of the actual observed data. Finally, the standard deviation of measurement errors for water production rate is assumed equal to 1.0 m^3/day for all six producers, the value of observed water production rate are small and we decided to adopted a minimum value of 1.0 m^3/day for the standard deviation. With these data, one can construct the measurement error covariance matrix.

The reservoir model parameters vector, $\boldsymbol{m}$, has dimension given by $N_m = 3,522$ as noted earlier. As described earlier in this dissertation, to promote computational efficiency, we reduce the order of the input space vector by applying the PCA method to the prior correlation matrix. The results are presented in Fig. 4.30, in which we present the first 300 singular values of the prior correlation matrix. For a $p_{N_v} = 0.9966$, see the discussion preceding Eq. 3.42, we need to keep only 162 singular values, i.e., $N_v = 162$, see the discussion preceding Eq. 3.40. The vertical dashed line in Fig. 4.30 marks the 162nd singular value position.

For the PUNQ-S3 case, we chose to solve a total of $N_e = 250$ minimization problems to find modes of the posterior pdf. The $N_e = 250$ initial guesses are sampled from the prior pdf using LHS [74]. To form the initial training set, an additional of $N_a = 750$ models are also sampled from the prior pdf using LHS, resulting in an initial training set with $N_t = 1,000$ training examples.

Similar to Gao et al. [37, 38], we opted to condition all $N_t = 1,000$ selected models to permeability hard data available from the drilled wells. In this dissertation, we use the RML method [87, 64] to condition the model to the hard data, represented here by "measurements" of log-permeabilities at gridblocks penetrated by wells. Therefore, we treat the hard data

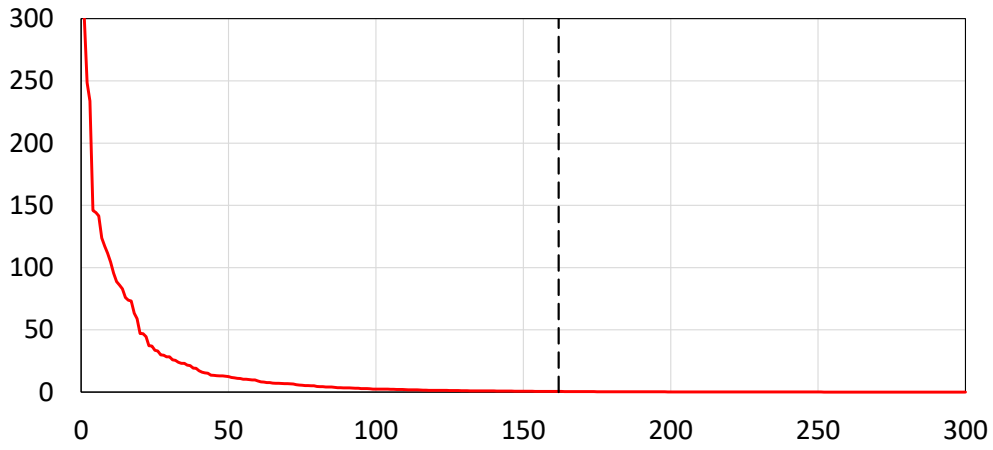


Figure 4.30: The SVD results of the prior correlation matrix for the PUNQ-S3 case: the red curve represents the first 300 singular values, the dashed vertical line represents the position of the 162nd singular value.

as a static observed data for the reservoir permeability fields. In this case, the relationship between the vector of reservoir parameters, $\boldsymbol{m}$, and the hard data is linear, and is given by

$$\boldsymbol{h}_{\text{hd}} = G_{\text{hd}}\, \boldsymbol{m}\,. \tag{4.19}$$

In Eq. 4.19, $\boldsymbol{h}_{\text{hd}}$ represents the $N_{\text{hd}}-$dimensional column vector of hard data, i.e., a total of N_{hd} hard data values are observed, and G_{hd} represents the $N_{\text{hd}} \times N_m$ sensitivity matrix for the hard data. The ℓth row of the matrix G_{hd}, for $\ell = 1, \ldots, N_{\text{hd}}$, has all entries equal to zero, except for its jth entry which has a value equal to 1. In this context, the jth entry of the ℓth row (note that j can assume the value $j = 1, \ldots, N_m$) represents the position of the ℓth hard data, which should correspond to the jth entry on the vector of reservoir parameters $\boldsymbol{m}$.

With the linear relationship given by Eq. 4.19, the kth selected initial model $\boldsymbol{m}_k$, for $k = 1, \ldots, N_t = 1,000$, is conditioned to the available hard data using the following equation from Oliver et al. [89]

$$\boldsymbol{m}_{k,\text{hd}} = \boldsymbol{m}_k + C_M\, G_{\text{hd}}^T \Big[G_{\text{hd}}\, C_M\, G_{\text{hd}}^T + C_{\text{hd}} \Big]^{-1} \big(\boldsymbol{h}_{\text{hd,obs}} + \boldsymbol{\xi}_{k,\text{hd}}^{(\text{uc})} - G_{\text{hd}}\, \boldsymbol{m}_k \big)\,. \tag{4.20}$$

Well	PRO-01	PRO-04	PRO-05	PRO-11	PRO-12	PRO-15
Layer 1 - $\ln(k)$	3.6743	6.5140	6.0850	4.0519	3.5502	5.5903
Layer 1 - $\ln(k_z)$	2.2059	5.2298	6.1176	3.7693	3.0413	6.1981
Layer 2 - $\ln(k)$	3.2129	3.0137	2.5084	2.7883	4.2405	3.8122
Layer 2 - $\ln(k_z)$	1.2633	0.9875	1.0094	1.8578	3.4015	2.4284
Layer 3 - $\ln(k)$	4.6476	3.8683	5.7408	6.2875	4.2277	4.7780
Layer 3 - $\ln(k_z)$	2.5500	3.3104	5.4433	5.5541	3.2929	4.6214
Layer 4 - $\ln(k)$	5.5268	6.5330	4.8816	4.9159	5.6424	6.1363
Layer 4 - $\ln(k_z)$	4.3646	3.6502	4.1680	3.1715	3.5704	3.8065
Layer 5 - $\ln(k)$	5.4004	6.1618	3.0654	4.8846	6.0195	7.3068
Layer 5 - $\ln(k_z)$	5.6233	5.9546	2.5284	2.9698	5.5591	6.0218

Table 4.5: Actual observed hard data considered for the PUNQ-S3 case.

In Eq. 4.20, $\boldsymbol{m}_{k,\mathrm{hd}}$, for $k = 1, \ldots, N_t = 1,000$, denotes the kth conditioned model, $\boldsymbol{h}_{\mathrm{hd,obs}}$ represents the vector of actual available hard data, and C_{hd} represents the measurement errors covariance matrix for the hard data. In this dissertation, we construct a diagonal matrix C_{hd} by considering the standard deviation of the ℓth hard data equal to 20% of the true log-permeability in the grid block corresponding to the hard data location. Also in Eq. 4.20, the $N_{\mathrm{hd}}-$dimensional column vector $\boldsymbol{\xi}^{(\mathrm{uc})}_{k,\mathrm{hd}}$, for $k = 1, \ldots, N_t = 1,000$, represents a perturbation to the vector of actual hard data $\boldsymbol{h}_{\mathrm{hd,obs}}$. Each vector $\boldsymbol{\xi}^{(\mathrm{uc})}_{k,\mathrm{hd}}$, for $k = 1, \ldots, N_t = 1,000$, is obtained by sampling a Gaussian distribution with zero mean and covariance matrix equal to C_{hd} [89]. The hard data used to conditioned the models are the same as those developed by Gao et al. [37, 38]. These data consist of the horizontal and vertical gridblock log-permeability for all five layers at each producer location, i.e., the five vertical gridblocks penetrated by each vertical producer. The hard data which compose the vector $\boldsymbol{h}_{\mathrm{hd,obs}}$ are presented in Table 4.5.

To construct the initial training set, we run the reservoir simulator for each one of the conditioned initial models $\boldsymbol{m}_{k,\mathrm{hd}}$, for $k = 1, \ldots, N_t = 1,000$. Then, the conditioned initial models are transformed to their reduced order counterparts, see Eq. 3.47, using

$$\boldsymbol{z}_k = W_{N_v}^{-1/2}\, U_{N_v}^T\, S_M^{-1}\, (\boldsymbol{m}_{k,\mathrm{hd}} - \boldsymbol{m}_{\mathrm{pr}})\,, \qquad \text{for} \quad k = 1, 2, \ldots, N_t\,. \tag{4.21}$$

The initial training set is then constructed using the vectors $\boldsymbol{z}_k$, for $k = 1, \ldots, N_t = 1,000$, and the associated reservoir predictions. It is worthwhile to mention that we only condition the initial models to the hard data. Subsequent models added to the training set are from the minimization steps and do not need to be conditioned to the hard data.

As explained earlier, we conduct the minimization problems in the reduced order space of the vector $\boldsymbol{z}$. When we need to add a new point to the training set, we convert the corresponding $\boldsymbol{z}$ to its respective $\boldsymbol{m}$, using Eq. 3.49, then we run the reservoir simulator. For the PUNQ-S3 case, we adopt a minimum distance to update the training set of $d_{\min} = 1.0\text{E}-5$, which seems to work well when the distance is normalized using the dimension of the vector $\boldsymbol{z}$, as discussed earlier. The trust-region parameters are set to $\delta_e^{(1)} = 0.2$, for $e = 1, \ldots, N_e$, $\delta_{\min} = 2.0\text{E}-5$ and $\delta_{\max} = 0.5$, in this case, the reduced order vector $\boldsymbol{z}$ resembles the dimensionless vector $\hat{\boldsymbol{m}}$, see the discussion preceding Eq. 2.11, thus similar numerical values for the trust region minimization problem are used as in the previous example. The convergence criteria are set to $\epsilon_1 = \epsilon_2 = 1.0\text{E}-6$ and $p_{\text{conv}} = 0.90$, so a minimum of 90% of the minimization problems have to converge. As before, numerical experiments using the $N_e = 250$ selected models as a test set and using the $N_a = 750$ models to construct a training set are conducted to determine the LS-SVR proxy parameters, which results in a $\gamma = 800$ and $\sigma = 4.1$, corresponding to a $c_\sigma = 0.32$, see Eq. 3.9. During the minimization loop, we set $N_{\text{cut}} = 4,000$ and $O_{N,\text{cut}} = 80.0$, which means that when the training set becomes larger than 4,000 training examples, then at each iteration of the minimization loop, we remove from the training set the training examples which provide normalized objective function values greater than 80.0, see the discussion preceding Eq. 3.27.

In Fig. 4.31 we present the minimization loop results. As in the previous cases, the final values of the normalized objective function obtained after convergence are about four orders of magnitude lower than the corresponding values for the prior models. Once more, this corroborates that the LS-SVR proxy gradient is indeed a good alternative to the adjoint solution. However, the Eq. 3.27, which gives the approximate upper bound for the value of the normalized objective function at the minima, provides an upper bound of 1.17, and the

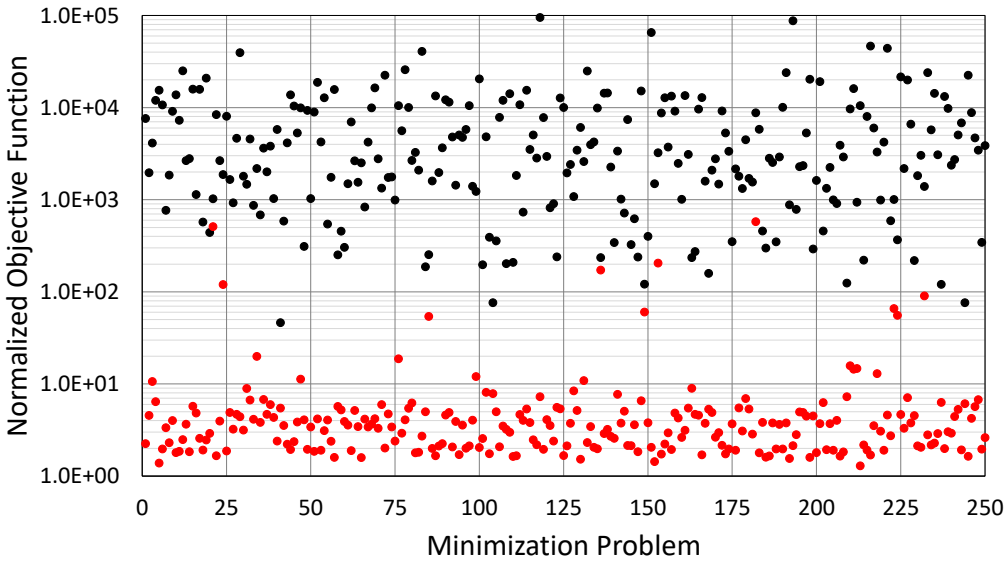


Figure 4.31: Minimization loop results for the PUNQ-S3 case: prior models (black dots), converged models (red dots).

minimum value obtained in the minimization loop is 1.29. The performance for the LS-SVR gradient for this case is slightly inferior to the performance observed in the one-dimensional case. More than 50 models present normalized objective function values less than 2.00.

At the end of the minimization loop, a total of 10,738 reservoir simulation runs were required in order for 90% of the 250 minimization problems to converge. Furthermore, this resulted in a training set with 9,454 training examples. Using a value of $O_{N\max} = 20.0$, the size of the training set is reduced to a total of 9,409 training examples, which is still a large training set. To further reduce the training set size, we applied the seed procedure described in Subsection 3.2.1 of Chapter 3. To do so, we selected the modes found which give normalized objective function values less than 8.0, which represent 226 modes, to start the seed for a new training set. Then, we compare the models from the training set with the seed models using a normalized distance of $2.0\mathrm{E}-3$; for the cases where the distance for the models in the seed is larger than the adopted distance, the model is added to the seed. At the end, 6,326 training examples were selected to form the final training set. Then, we use this final training set to train the final LS-SVR proxy model.

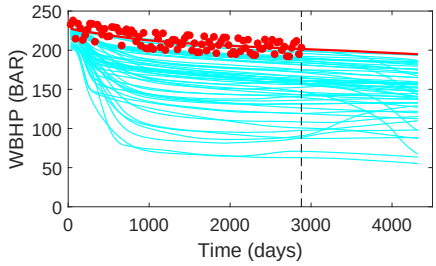
From the 250 minimization problems, 54 converge to modes that give a normalized objective function value less than 2.0 and these models are used to construct the GMM

approximation. In Figs. 4.32 through 4.37 we present the history matching results for those 54 selected modes. The original prior predictions for the 54 models are compared to the predictions after history matching. As one can see, the overall history matching performance for these 54 selected model is reasonable. In all figures, the solid thick red curve represents the true value, the red dots represent the observed data, and the cyan curves represents the corresponding model predictions.

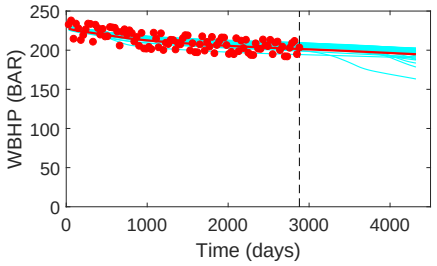
In Figs. 4.38 through 4.49 we compare the prior log-permeability horizontal and vertical permeability fields with the corresponding conditioned models after history matching for the first six selected modes. As one can see, although we conducted the minimization problems in the reduced order space of the vector $\boldsymbol{z}$, in which we kept only 162 principal directions, the overall minimization results present permeability fields which are consistent with the prior models, i.e., the minimization process preserved the geological information.

As before, the selected 54 modes are clustered into $N_c = 25$ clusters using the k-medoids clustering algorithm [63]. Then, the clusters are used to build a GMM approximation for the posterior pdf. Again, the weights in the GMM approximation are all set to $w_\ell = 1/25 = 0.04$ for $\ell = 1, \ldots, N_c$. Using this GMM approximation as the proposal distribution, we run five chains with a total of 100,000 states each. As before, we first run short chains of length 2,000 states to tune the parameter μ_γ. For this case, we selected a parameter $\mu_\gamma = 35.0$, leading to a learning rate of $\gamma_i = 1.33\text{E} - 3$, which results in an acceptance rate of 26.4%. The convergence of the five chains based on the MPSRF method [14] is presented in Fig. 4.50. As one can see, the MPSRF reaches unity after 20,000 states are generated. Therefore, from state 20,000 onward, MPSRF indicates that the chains have converged to the posterior pdf, which is a fast convergence, as before.

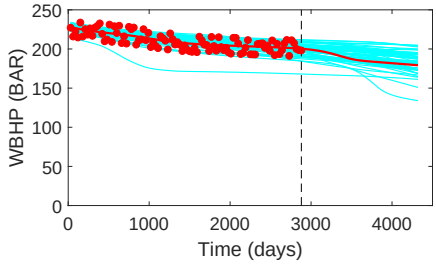
In the Figs. 4.51 and 4.52 the posterior mean for, respectively, horizontal and vertical gridblock log-permeability fields, which are obtained using our proposed methodology, are compared to the respective true log-permeability fields. As is well known, the posterior mean is too smooth to give a realization that shows detailed geological features like those shown in the posterior log-permeability fields of Figs. 4.38 through 4.49, although, one can see that



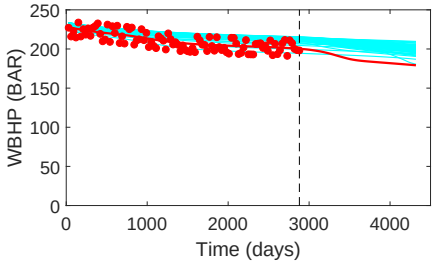
(a) BHP PRO-01: before history matching.



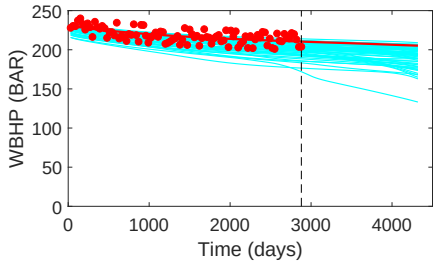
(b) BHP PRO-01: after history matching.



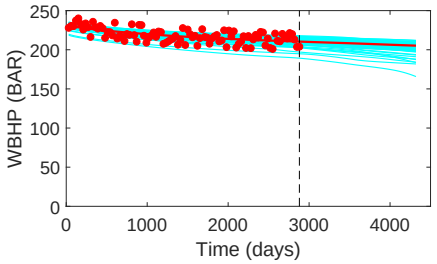
(c) BHP PRO-04: before history matching.



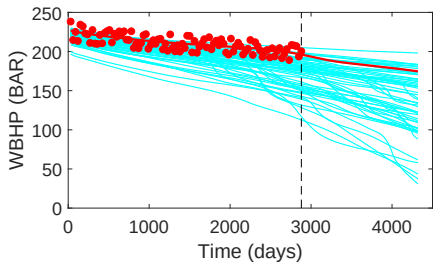
(d) BHP PRO-04: after history matching.



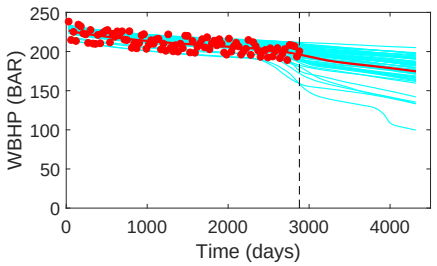
(e) BHP PRO-05: before history matching.



(f) BHP PRO-05: after history matching.

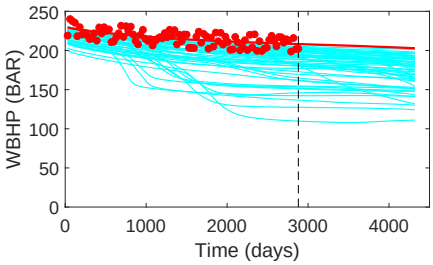


(g) BHP PRO-11: before history matching.

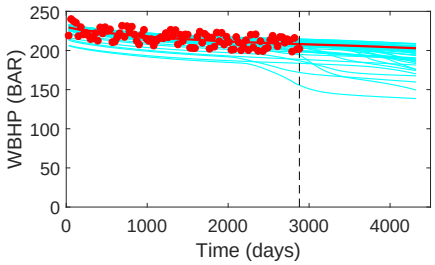


(h) BHP PRO-11: after history matching.

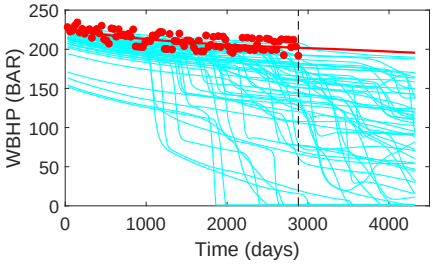
Figure 4.32: History matching results for the PUNQ-S3 case: well bottom-hole pressure at producers PRO-01, PRO-04, PRO-05, and PRO-11. A comparison between the original prior models predictions (left column) with the predictions after the history matching (right column) is presented. In all figures, we show the corresponding true value (solid thick red curve), the observed data (red dots), and the models predictions (cyan curves).



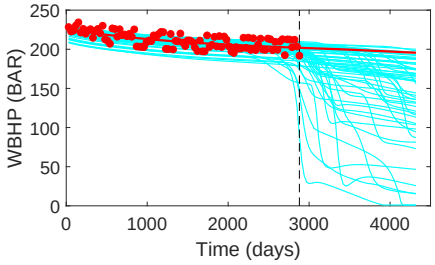
(a) BHP PRO-12: before history matching.



(b) BHP PRO-12: after history matching.

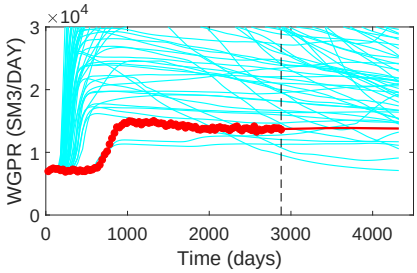


(c) BHP PRO-15: before history matching.

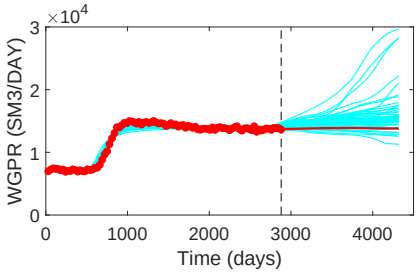


(d) BHP PRO-15: after history matching.

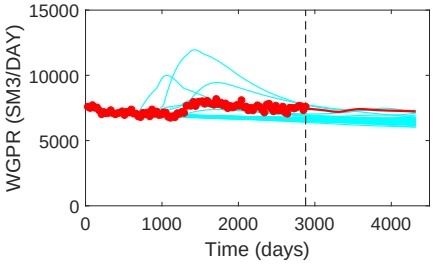
Figure 4.33: Same as in Fig. 4.32 for producers PRO-12 and PRO-15.



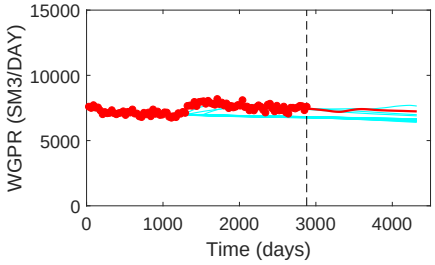
(a) GPR PRO-01: before history matching.



(b) GPR PRO-01: after history matching.

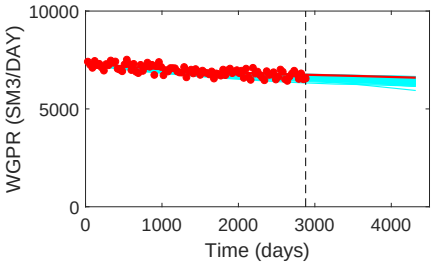


(c) GPR PRO-04: before history matching.

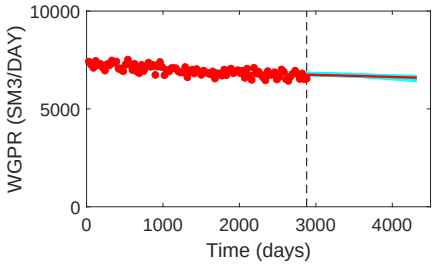


(d) GPR PRO-04: after history matching.

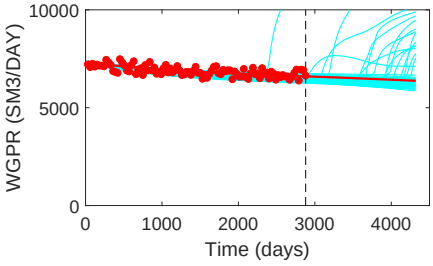
Figure 4.34: History matching results for the PUNQ-S3 case: gas production rate at producers PRO-01 and PRO-04. A comparison between the original prior models predictions (left column) with the predictions after the history matching (right column) is presented. In all figures, we show the corresponding true value (solid thick red curve), the observed data (red dots), and the models predictions (cyan curves).



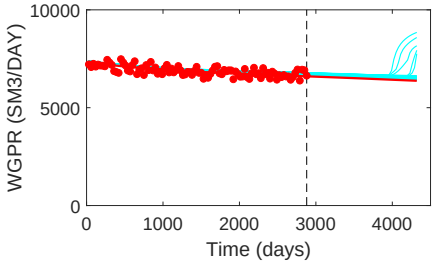
(a) GPR PRO-05: before history matching.



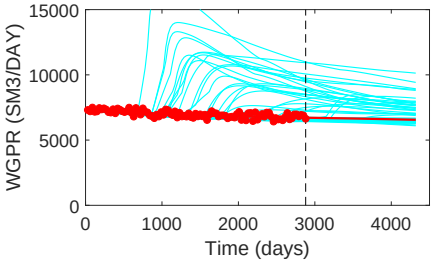
(b) GPR PRO-05: after history matching.



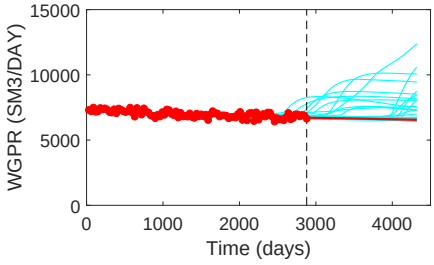
(c) GPR PRO-11: before history matching.



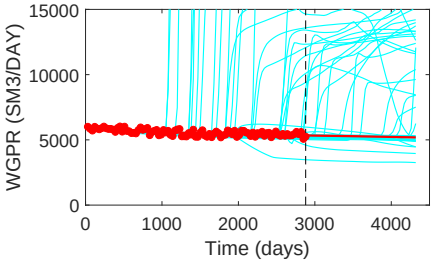
(d) GPR PRO-11: after history matching.



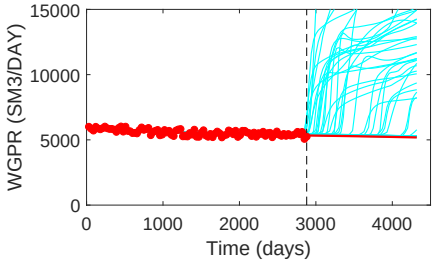
(e) GPR PRO-12: before history matching.



(f) GPR PRO-12: after history matching.

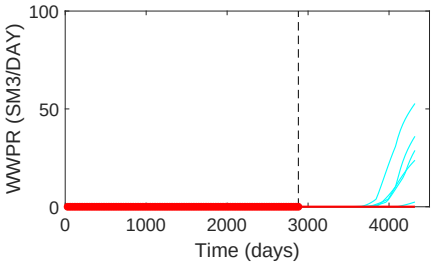


(g) GPR PRO-15: before history matching.

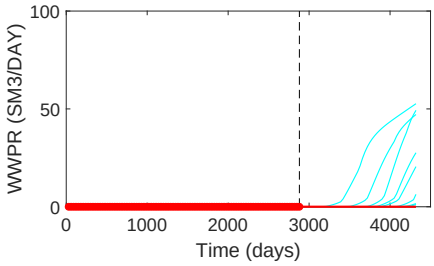


(h) GPR PRO-15: after history matching.

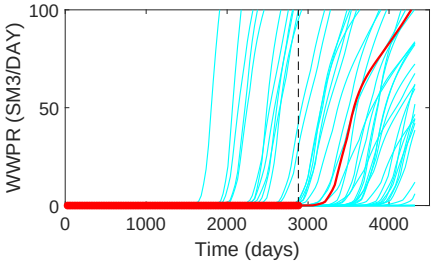
Figure 4.35: History matching results for the PUNQ-S3 case: gas production rate at producers PRO-05, PRO-11, PRO-12, and PRO-15. A comparison between the original prior models predictions (left column) with the predictions after the history matching (right column) is presented. In all figures, we show the corresponding true value (solid thick red curve), the observed data (red dots), and the models predictions (cyan curves).



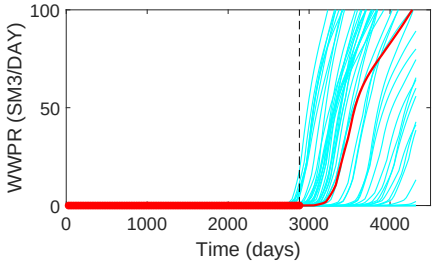
(a) WPR PRO-01: before history matching.



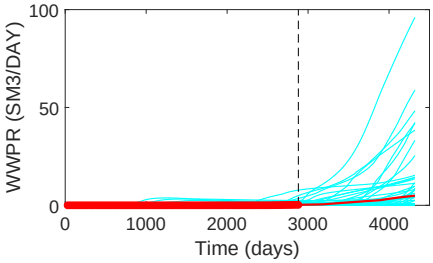
(b) WPR PRO-01: after history matching.



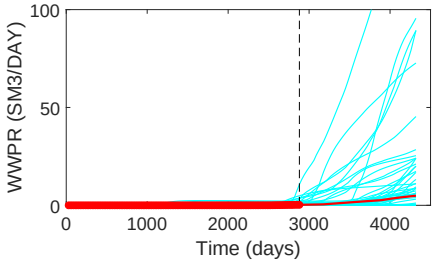
(c) WPR PRO-04: before history matching.



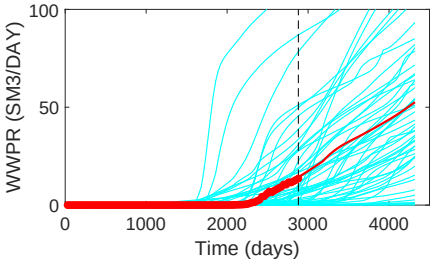
(d) WPR PRO-04: after history matching.



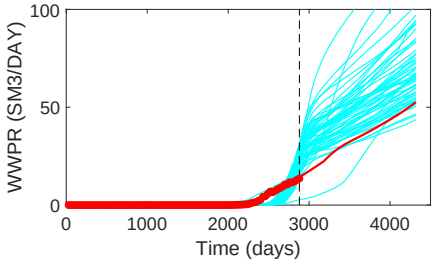
(e) WPR PRO-05: before history matching.



(f) WPR PRO-05: after history matching.

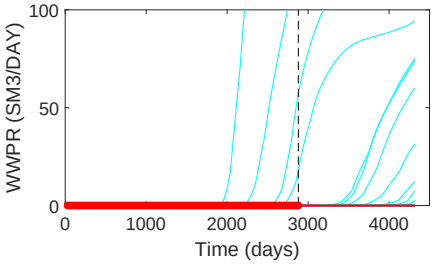


(g) WPR PRO-11: before history matching.

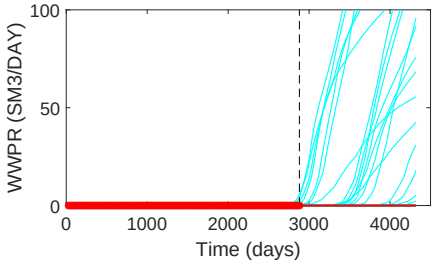


(h) WPR PRO-11: after history matching.

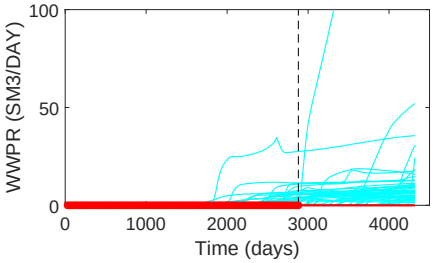
Figure 4.36: History matching results for the PUNQ-S3 case: water production rate at producers PRO-01, PRO-04, PRO-05, and PRO-11. A comparison between the original prior models predictions (left column) with the predictions after the history matching (right column) is presented. In all figures, we show the corresponding true value (solid thick red curve), the observed data (red dots), and the models predictions (cyan curves).



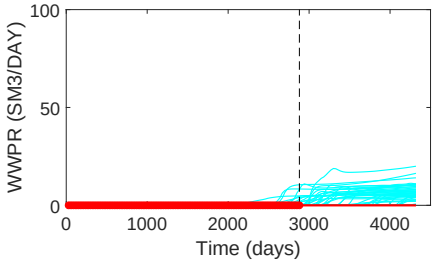
(a) WPR PRO-12: before history matching.



(b) WPR PRO-12: after history matching.



(c) WPR PRO-15: before history matching.



(d) WPR PRO-15: after history matching.

Figure 4.37: History matching results for the PUNQ-S3 case: water production rate at producers PRO-12 and PRO-15. A comparison between the original prior models predictions (left column) with the predictions after the history matching (right column) is presented. In all figures, we show the corresponding true value (solid thick red curve), the observed data (red dots), and the models predictions (cyan curves).

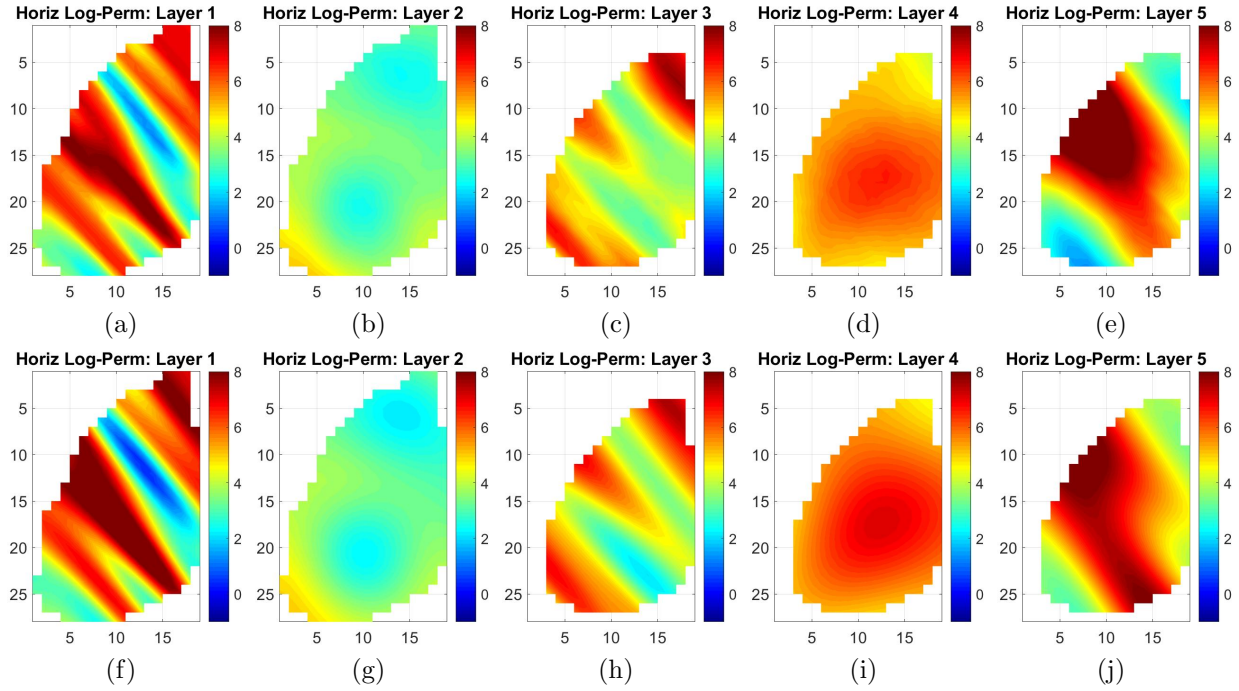


Figure 4.38: Horizontal log-permeability fields for the PUNQ-S3 case comparing the prior model (top row) with the history matched model (bottom row) for the converged model 1: (a)-(f) first layer, (b)-(g) second layer, (c)-(h) third layer, (d)-(i) fourth layer, and (e)-(j) fifth layer.

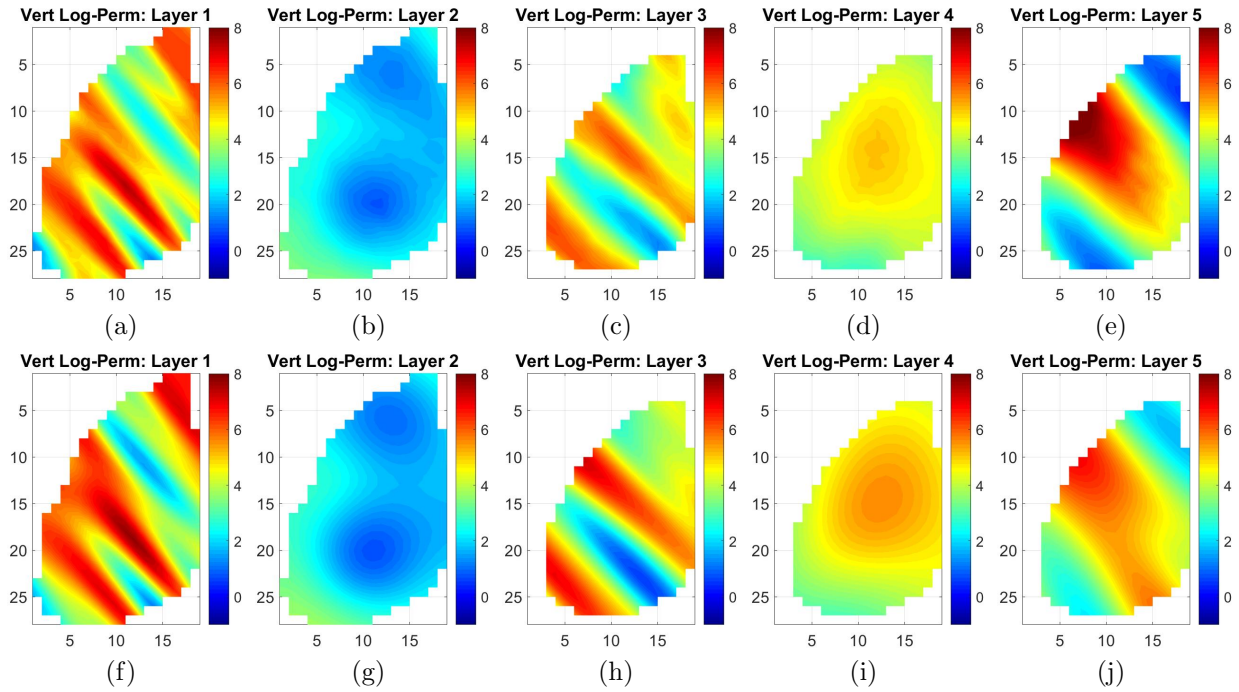


Figure 4.39: Vertical log-permeability fields for the PUNQ-S3 case comparing the prior model (top row) with the history matched model (bottom row) for the converged model 1: (a)-(f) first layer, (b)-(g) second layer, (c)-(h) third layer, (d)-(i) fourth layer, and (e)-(j) fifth layer.

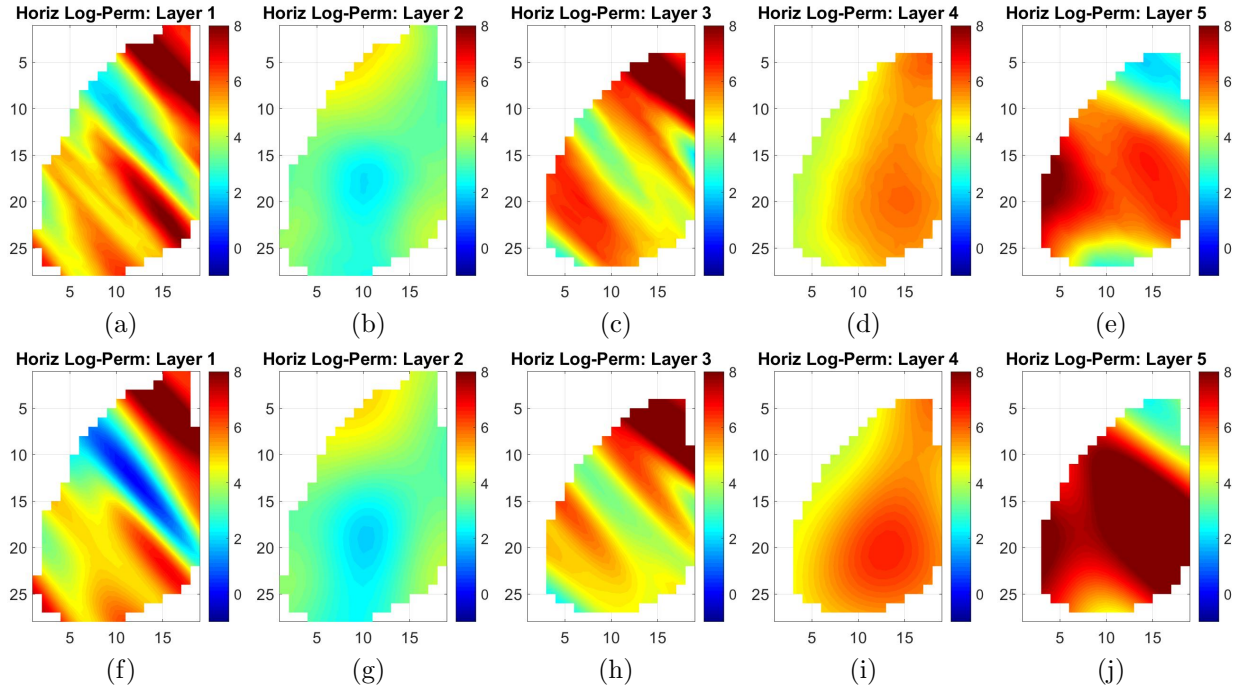


Figure 4.40: Horizontal log-permeability fields for the PUNQ-S3 case comparing the prior model (top row) with the history matched model (bottom row) for the converged model 2: (a)-(f) first layer, (b)-(g) second layer, (c)-(h) third layer, (d)-(i) fourth layer, and (e)-(j) fifth layer.

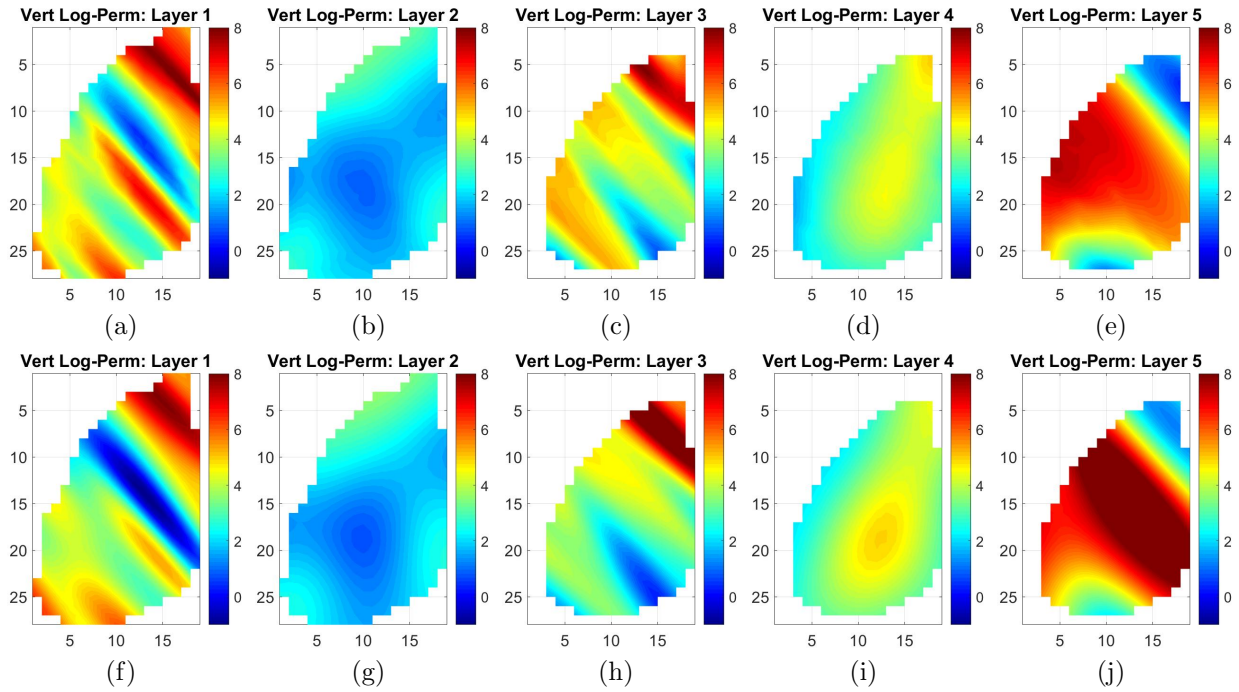


Figure 4.41: Vertical log-permeability fields for the PUNQ-S3 case comparing the prior model (top row) with the history matched model (bottom row) for the converged model 2: (a)-(f) first layer, (b)-(g) second layer, (c)-(h) third layer, (d)-(i) fourth layer, and (e)-(j) fifth layer.

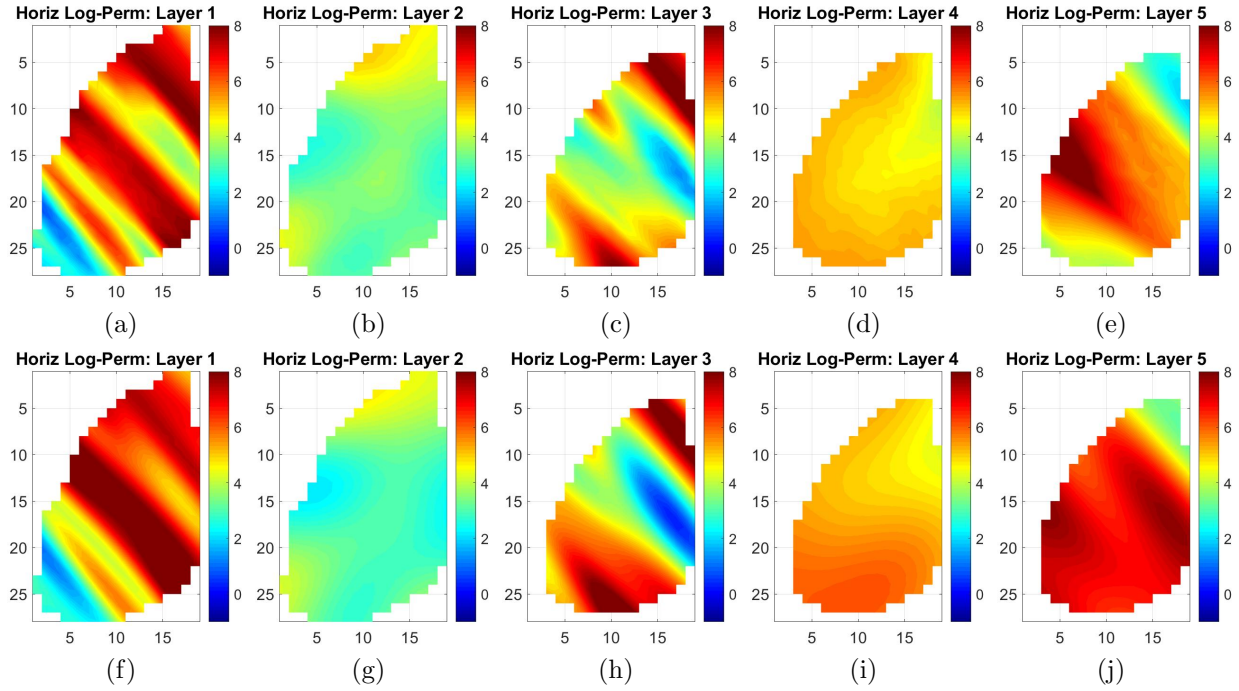


Figure 4.42: Horizontal log-permeability fields for the PUNQ-S3 case comparing the prior model (top row) with the history matched model (bottom row) for the converged model 3: (a)-(f) first layer, (b)-(g) second layer, (c)-(h) third layer, (d)-(i) fourth layer, and (e)-(j) fifth layer.

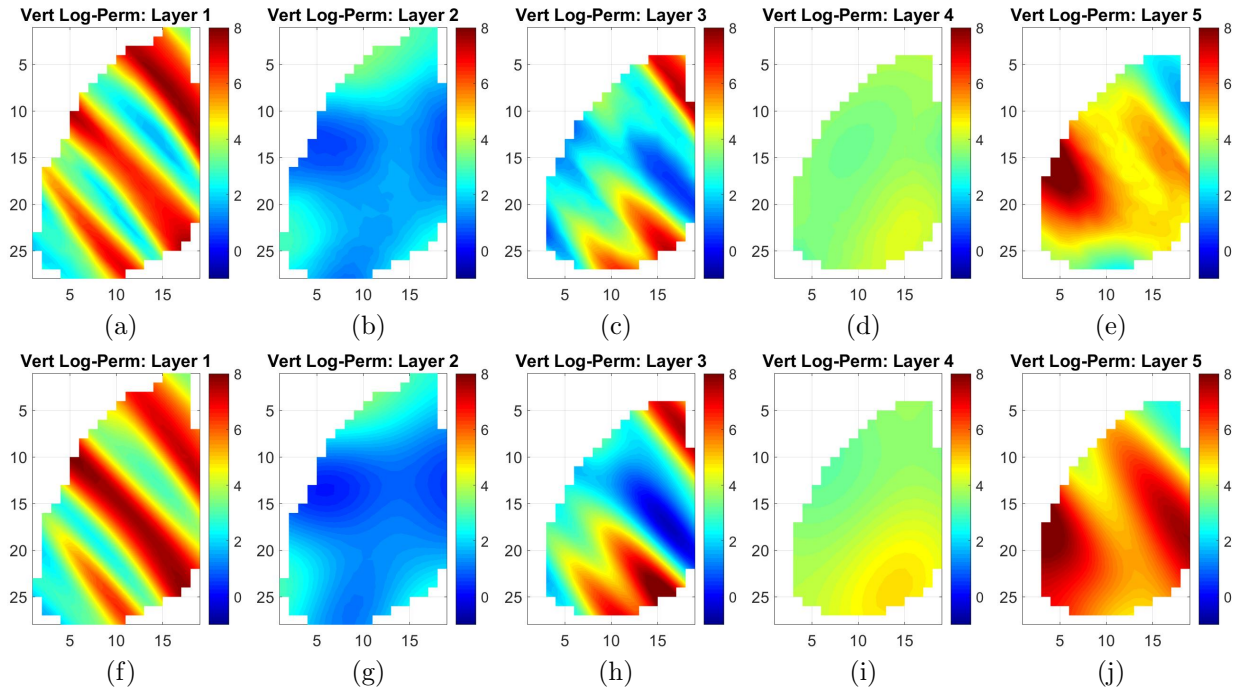


Figure 4.43: Vertical log-permeability fields for the PUNQ-S3 case comparing the prior model (top row) with the history matched model (bottom row) for the converged model 3: (a)-(f) first layer, (b)-(g) second layer, (c)-(h) third layer, (d)-(i) fourth layer, and (e)-(j) fifth layer.

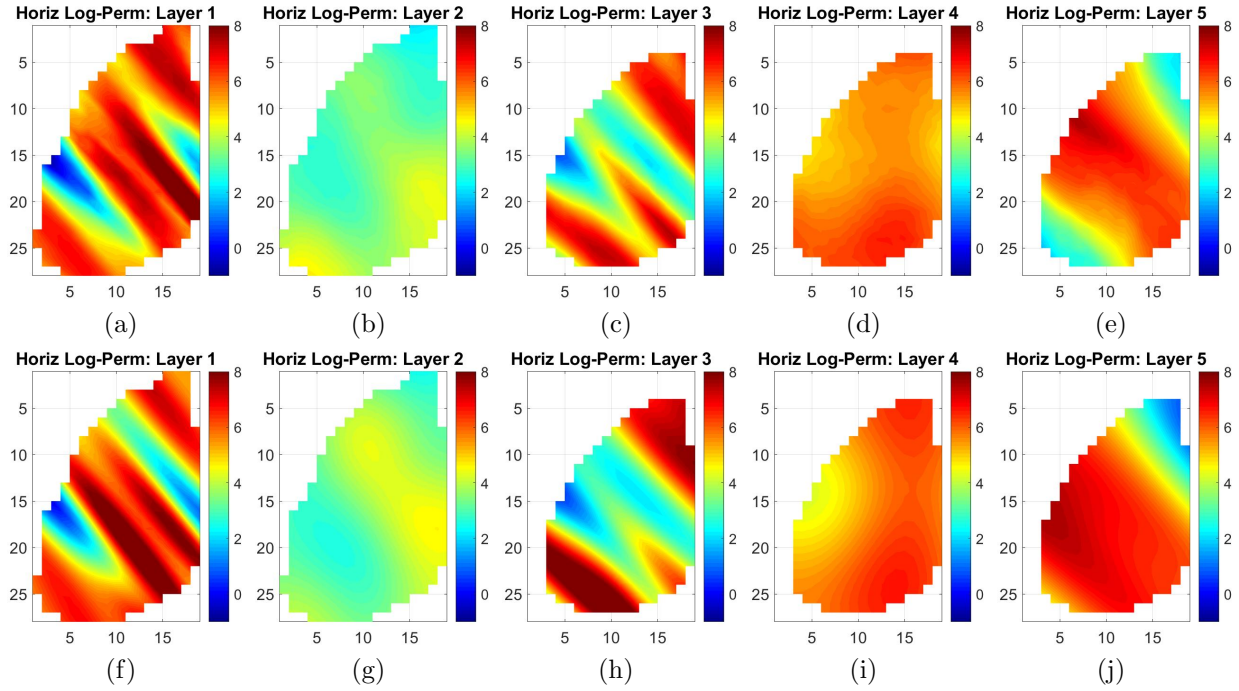


Figure 4.44: Horizontal log-permeability fields for the PUNQ-S3 case comparing the prior model (top row) with the history matched model (bottom row) for the converged model 4: (a)-(f) first layer, (b)-(g) second layer, (c)-(h) third layer, (d)-(i) fourth layer, and (e)-(j) fifth layer.

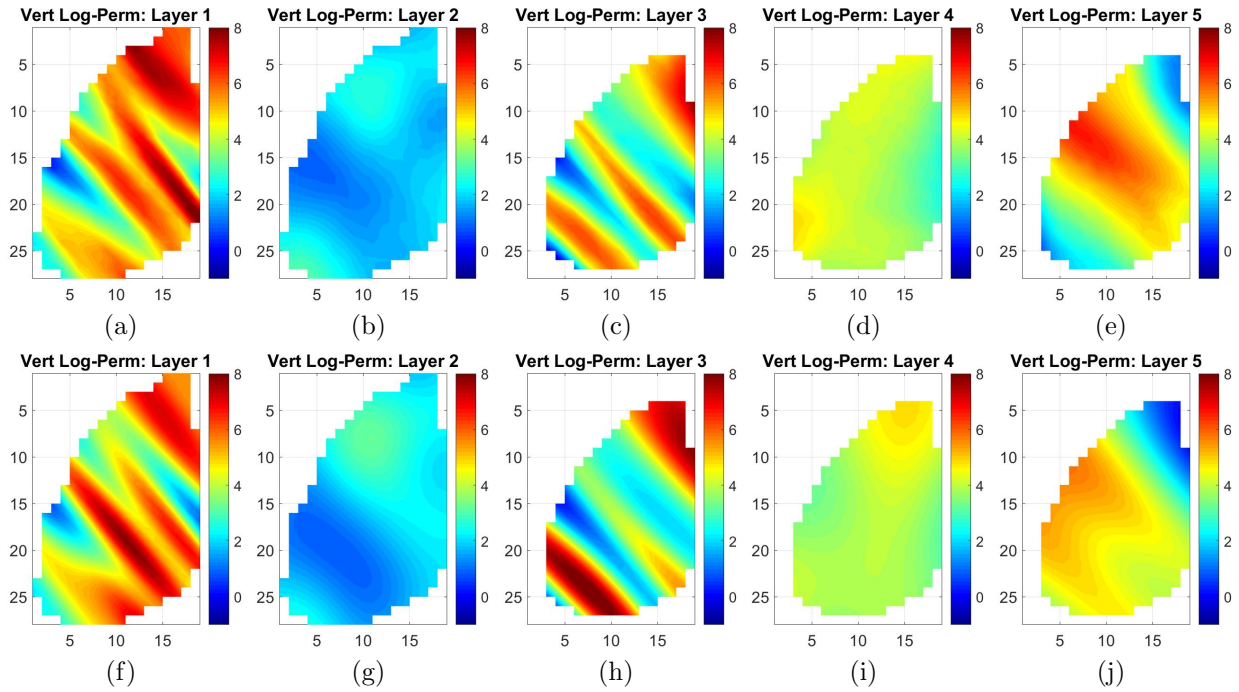


Figure 4.45: Vertical log-permeability fields for the PUNQ-S3 case comparing the prior model (top row) with the history matched model (bottom row) for the converged model 4: (a)-(f) first layer, (b)-(g) second layer, (c)-(h) third layer, (d)-(i) fourth layer, and (e)-(j) fifth layer.

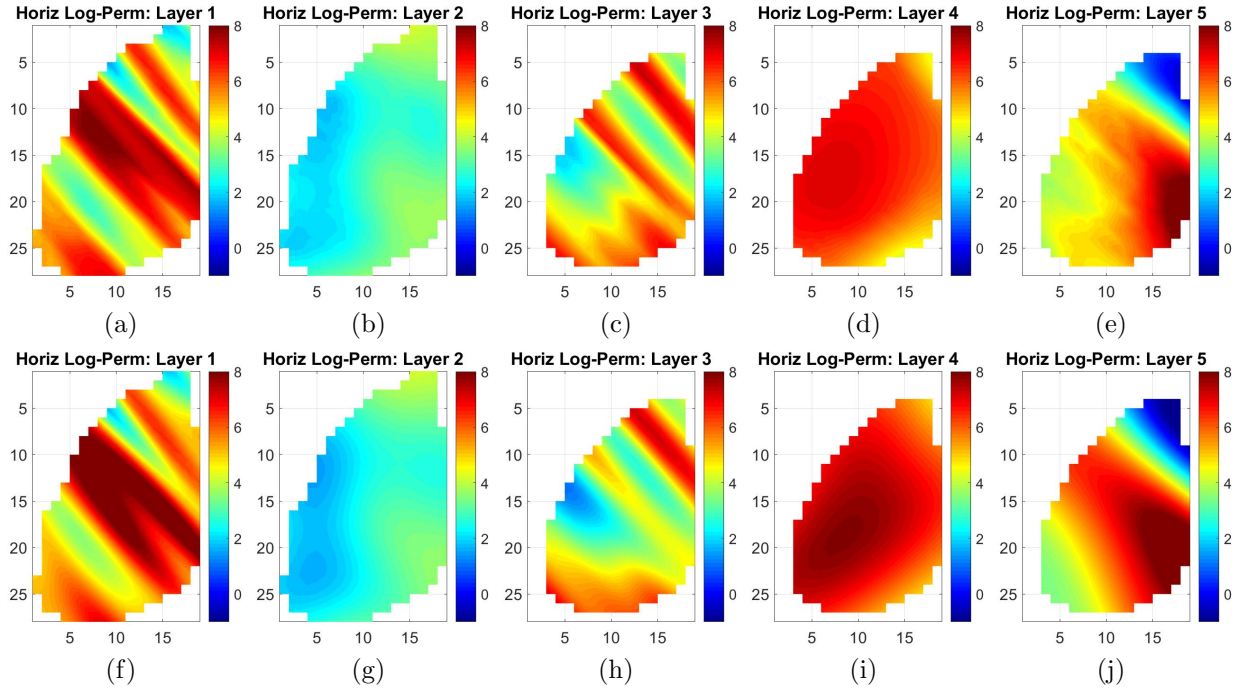


Figure 4.46: Horizontal log-permeability fields for the PUNQ-S3 case comparing the prior model (top row) with the history matched model (bottom row) for the converged model 5: (a)-(f) first layer, (b)-(g) second layer, (c)-(h) third layer, (d)-(i) fourth layer, and (e)-(j) fifth layer.

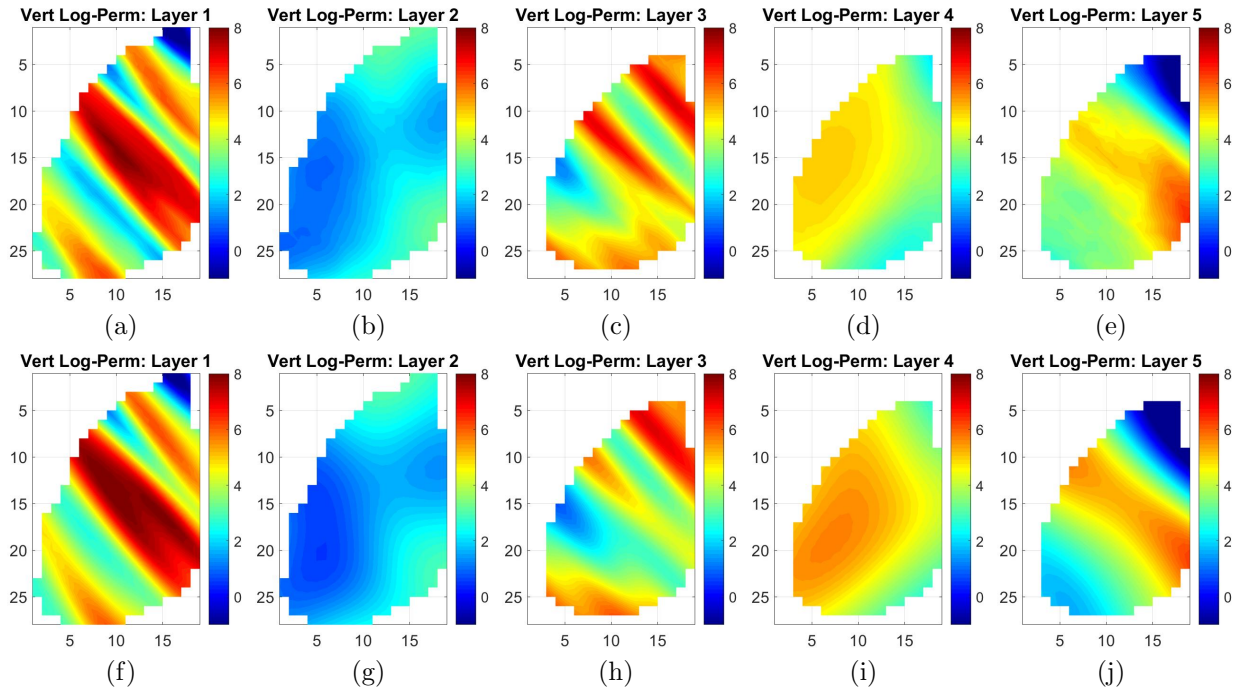


Figure 4.47: Vertical log-permeability fields for the PUNQ-S3 case comparing the prior model (top row) with the history matched model (bottom row) for the converged model 5: (a)-(f) first layer, (b)-(g) second layer, (c)-(h) third layer, (d)-(i) fourth layer, and (e)-(j) fifth layer.

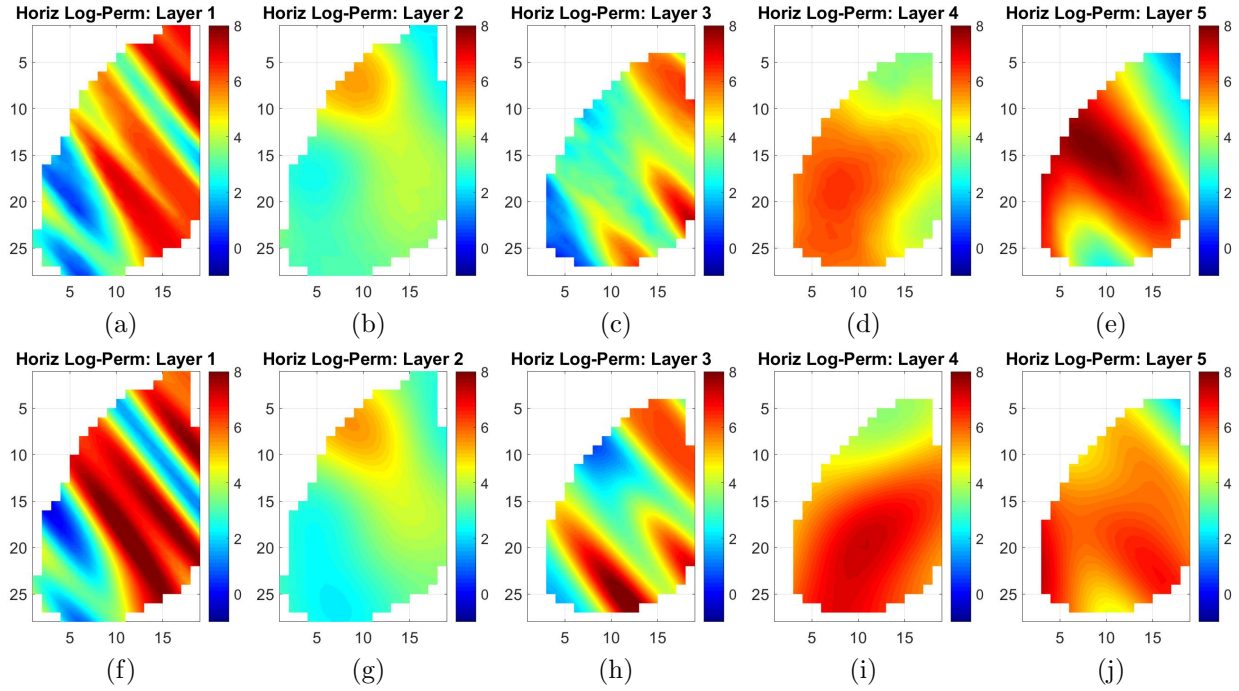


Figure 4.48: Horizontal log-permeability fields for the PUNQ-S3 case comparing the prior model (top row) with the history matched model (bottom row) for the converged model 6: (a)-(f) first layer, (b)-(g) second layer, (c)-(h) third layer, (d)-(i) fourth layer, and (e)-(j) fifth layer.

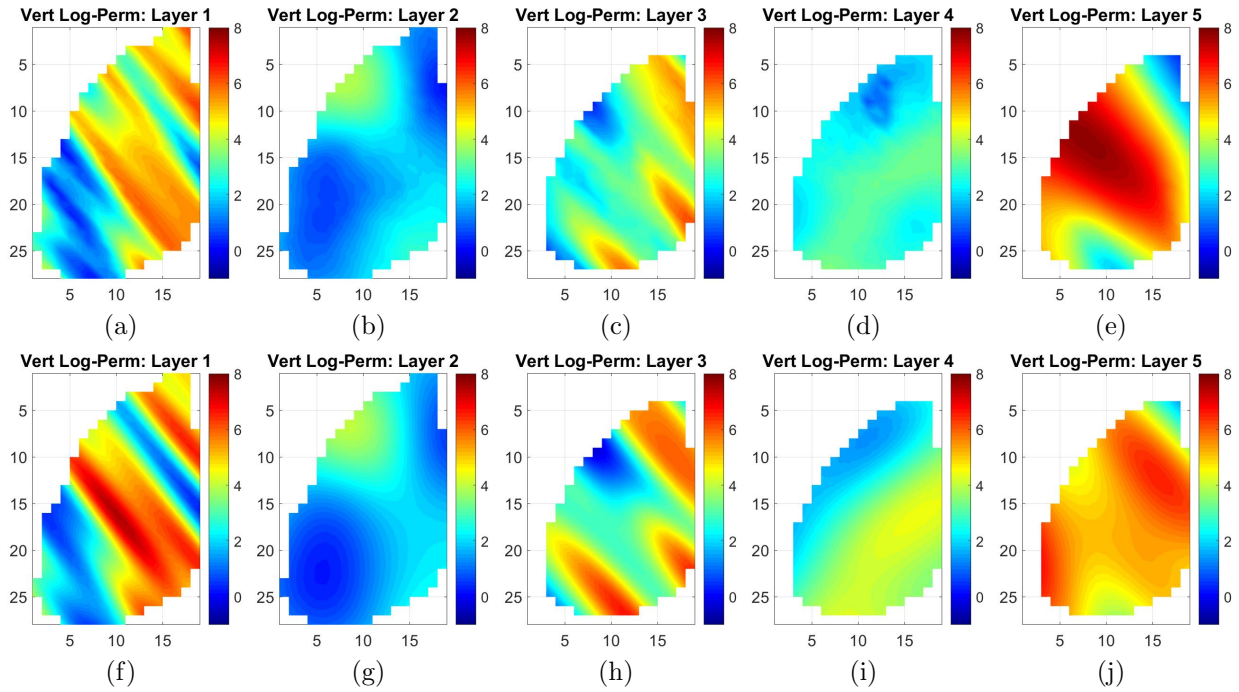


Figure 4.49: Vertical log-permeability fields for the PUNQ-S3 case comparing the prior model (top row) with the history matched model (bottom row) for the converged model 6: (a)-(f) first layer, (b)-(g) second layer, (c)-(h) third layer, (d)-(i) fourth layer, and (e)-(j) fifth layer.

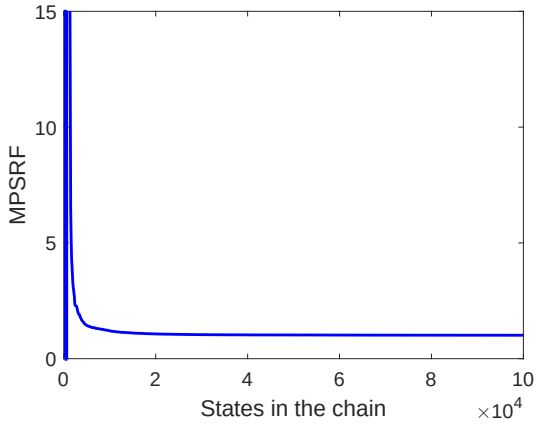


Figure 4.50: MCMC sampling convergence based on MPSRF for the PUNQ-S3 case.

some features of the true log-permeability field are represented in the posterior mean.

The uncertainty quantification results are presented in Figs. 4.53 through 4.55. To generate the results presented in Figs. 4.53 through 4.55, we reconstruct the marginal distributions using all states after state 40,000 for each one of the five chains, which results in a total of 300,000 states. The color code in all these figures is the same as described in Fig. 4.5, i.e., the solid thick black curve represents the mean of the marginal distribution, the two blue dashed curves represents the 25% (P25) and 75% (P75) percentiles, while the two solid thin black curves represent the 5% (P5) and 95% (P95) percentiles and, finally, the solid thick red curve represents the predictions using the true model and the red dots represent the observed data. The marginal distribution for well bottom hole pressure (WBHP) is presented in Fig. 4.53. The marginal distribution for well gas production rate (WGPR) is presented in Fig. 4.54. Finally, the marginal distribution for well water production rate (WWPR) is presented in Fig. 4.55. As one can see, the overall performance of the history matching is reasonably good, except for the gas production rates at producers PRO-01 and PRO-04. Although no base case is available for comparison, the uncertainty quantification results are plausible. One additional difficulty for this case is the fact that Gao et al. [37, 38] generate the true permeability field by sampling a probability distribution which was different from the prior pdf, resulting in a true case somehow different from the samples of the prior pdf.

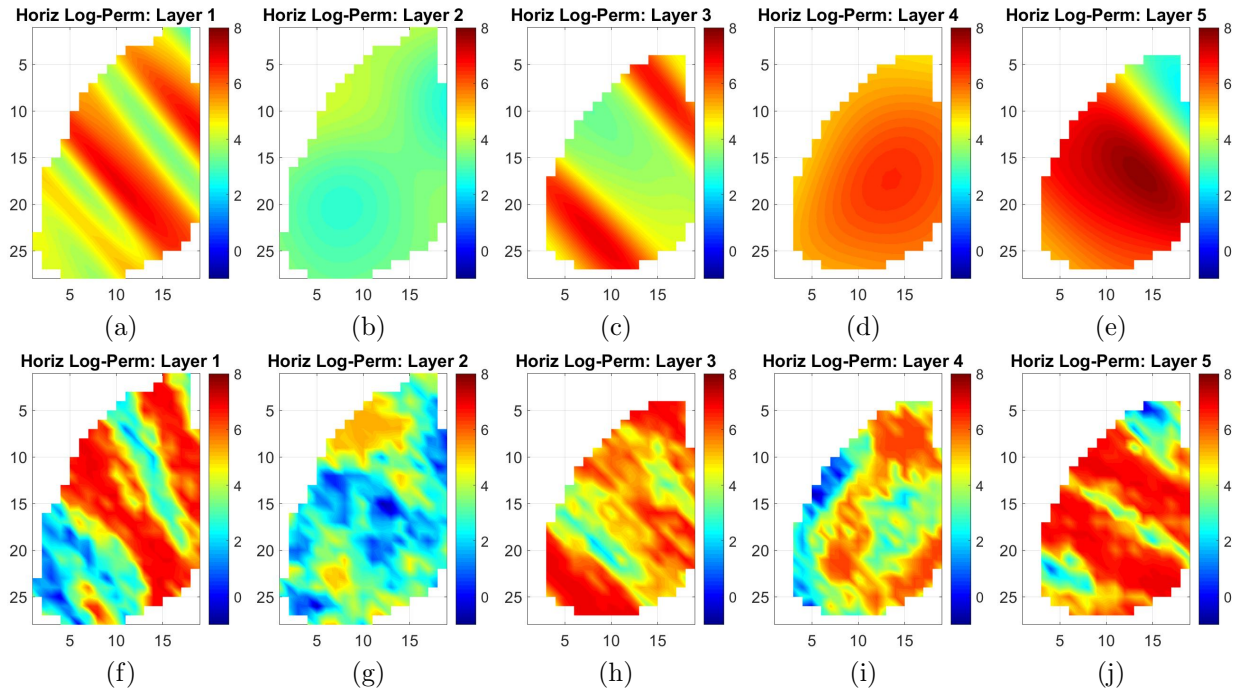


Figure 4.51: Posterior mean for horizontal log-permeability compared to the true horizontal log-permeability field: (a)-(e) shows, respectively, the first to fifth layers of the posterior mean, (f)-(j) shows, respectively, the first to fifth layers of the true permeability field.

4.5 Application to a Three-Dimensional Large Scale Reservoir Model Based on the PUNQ-S3 Case.

One of the main purposes of this research is to reduce the computational cost when quantifying uncertainty using the Metropolis-Hastings MCMC sampling algorithm, especially for large scale reservoir simulation models, i.e., cases where the reservoir parameter vector, $\boldsymbol{m}$, has a large dimension. However, the required data for applying the proposed methodology to a real large scale reservoir model is seldom available, since those models usually involves confidential and proprietary information.

To evaluate the performance of our proposed methodology when applied to a large scale problem, we decided to use the information from the PUNQ-S3 case to develop a large scale reservoir simulator model. We selected the PUNQ-S3 case since this particular model was constructed based on real reservoir information. The approach adopted to construct the large scale model was quite elementary, we simply divided each original gridblock into

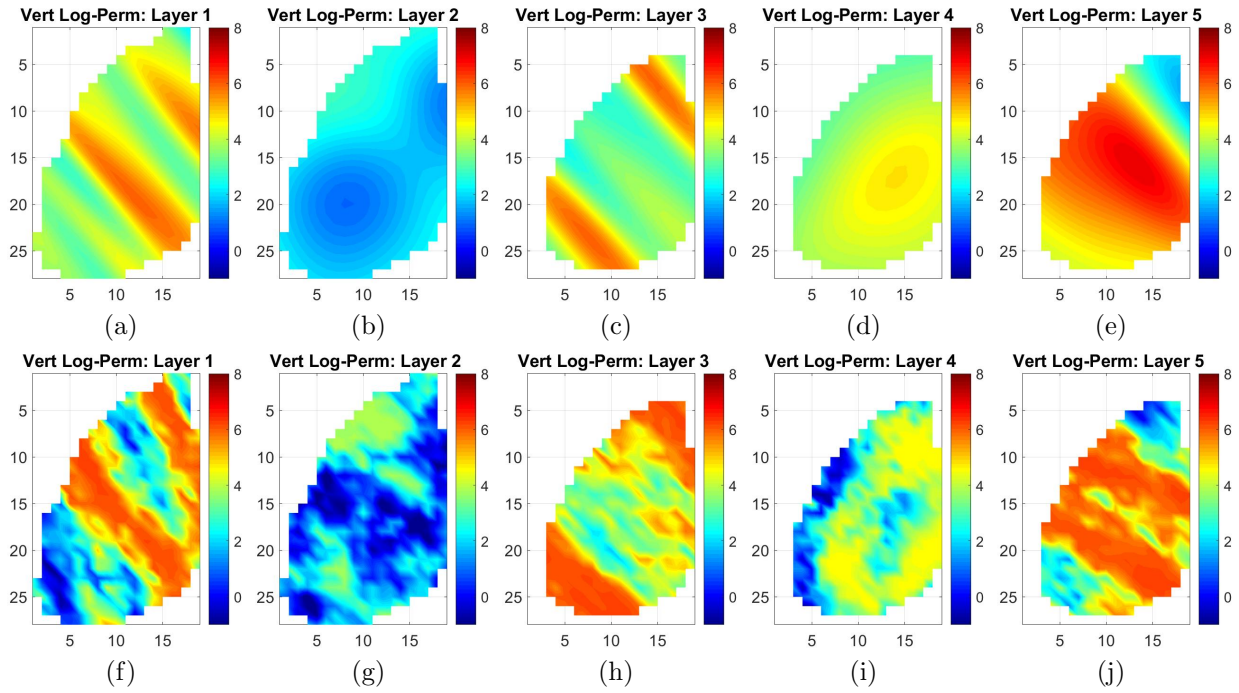


Figure 4.52: Posterior mean for vertical log-permeability compared to the true vertical log-permeability field: (a)-(e) shows, respectively, the first to fifth layers of the posterior mean, (f)-(j) shows, respectively, the first to fifth layers of the true permeability field.

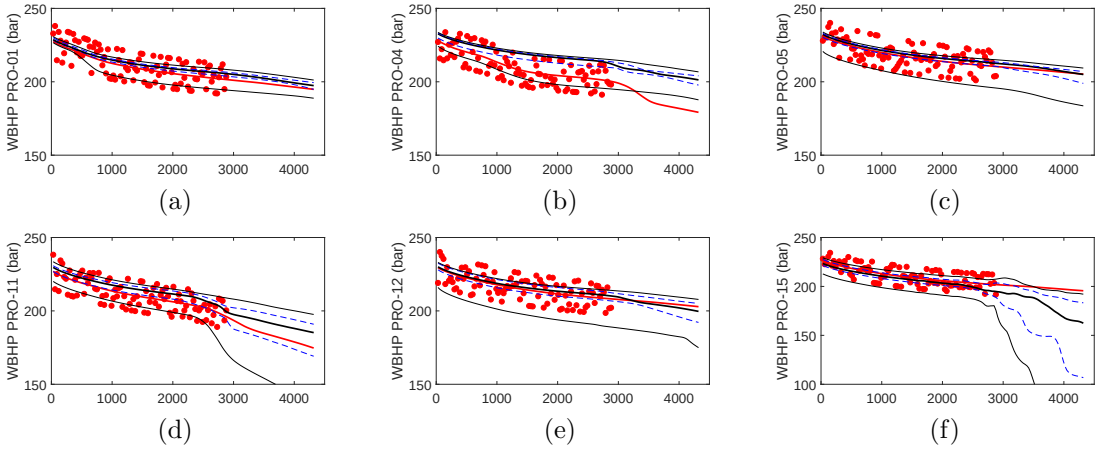


Figure 4.53: Marginal distribution for well bottom hole pressure for producers PRO-01 through PRO-15. In all figures, we show the corresponding true value (solid thick red curve), the mean of the marginal distribution (solid thick black curve), the percentiles P25 and P75 of the marginal distribution (two blue dashed curves), and the percentiles P5 and P95 of the marginal distribution (two thin black curves). To generate the marginal distributions presented here, all states after state 40,000 in the five chains are used, which gives a total of 300,000 states.

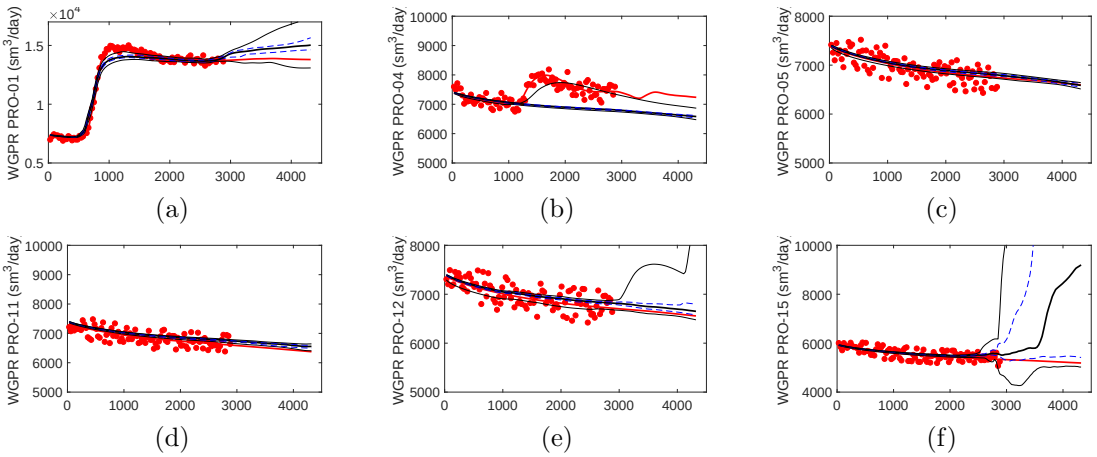


Figure 4.54: Marginal distribution for well gas production rate for producers PRO-01 through PRO-15. In all figures, we show the corresponding true value (solid thick red curve), the mean of the marginal distribution (solid thick black curve), the percentiles P25 and P75 of the marginal distribution (two blue dashed curves), and the percentiles P5 and P95 of the marginal distribution (two thin black curves). To generate the marginal distributions presented here, all states after state 40,000 in the five chains are used, which gives a total of 300,000 states.

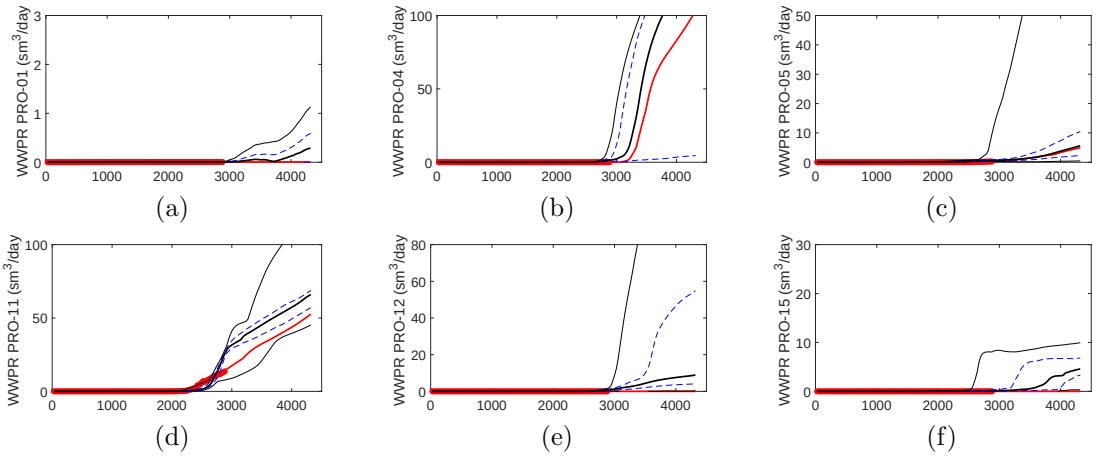


Figure 4.55: Marginal distribution for well water production rate for producers PRO-01 through PRO-15. In all figures, we show the corresponding true value (solid thick red curve), the mean of the marginal distribution (solid thick black curve), the percentiles P25 and P75 of the marginal distribution (two blue dashed curves), and the percentiles P5 and P95 of the marginal distribution (two thin black curves). To generate the marginal distributions presented here, all states after state 40,000 in the five chains are used, which gives a total of 300,000 states.

$3 \times 3 \times 3 = 27$ gridblocks. The original geometry of the PUNQ-S3 model is thus conserved. The resulting reservoir model has a total of $57 \times 84 \times 15 = 71,820$ gridblocks, with a total of 47,547 active gridblocks. As in the previous case, the reservoir parameters considered for history matching are the horizontal and vertical gridblock log-permeability. Consequently, one gets two parameters for each active gridblock, which results in a total of 95,094 history-matching reservoir parameters. The refined model with 95,094 parameters represents a limit for the computational power available for conducting this research. That is the main reason the applied refinement was selected, and no further refinement was sought.

Although all required information is available, for 95,094 parameters we were unable to assemble the $95,094 \times 95,094$ prior covariance matrix due to computational restrictions. The prior covariance matrix is necessary to generate the sample from the prior pdf. Nevertheless, in a real large scale reservoir problem, one is seldom able to assemble the prior covariance matrix in order to generate samples. Usually, the best scenario for a history matching problem is to have a finite number of samples, which are normally generated from a geological model. Therefore, to mimic a real problem, we do not need the prior covariance matrix, we just need a suitable way of generating samples from the prior pdf. To achieve this, we chose instead to generate samples from the coarse original PUNQ-S3 geological model, which we have available, then convert the generated samples to the refined grid. This approach somehow mimics a real problem, since normally the only available prior information about the reservoir parameters is presented in the form of a finite set of reservoir models. Naturally, simply converting the coarse samples to the refined grid will result in repeated reservoir properties within the refined gridblocks which correspond to the same original coarse gridblock. For this reason, we chose to add white noise to the resulting refined samples. The standard deviation for the white noise was selected as 5% of the actual gridblock log-permeability.

Assuming that a total of N_s samples are generated as described above, one can ap-

proximate the prior covariance matrix, C_M, using

$$C_M \cong \frac{1}{N_s - 1}\, \Delta M\, \Delta M^T\, . \tag{4.22}$$

In Eq. 4.22, each column of the $N_m \times N_s$ matrix ΔM represents a different centered sample, which is obtained by subtracting the sample mean from each individual sample, i.e., letting $[\Delta M]_s$ denote the sth column of the matrix ΔM, for $s = 1,\, 2,\, \cdots,\, N_s$, and $\boldsymbol{m}_s$ represent the sth generated samples, for $s = 1,\, 2,\, \cdots,\, N_s$, one gets

$$[\Delta M]_s = \boldsymbol{m}_s - \frac{1}{N_s} \sum_{s=1}^{N_s} \boldsymbol{m}_s\, , \qquad \text{for} \quad s = 1,\, 2,\, \ldots,\, N_s\, . \tag{4.23}$$

Notice that normally $N_s \ll N_m$, hence one is able to compute and assemble the matrix ΔM. With our available computational power, we cannot compute the matrix product $\Delta M\, \Delta M^T$. However, it is not necessary, as will become clear in the explanation below.

From Eq. 4.22, we compute the corresponding prior correlation matrix using

$$\widetilde{C}_M = S_M^{-1}\, C_M\, S_M^{-1} \cong \frac{1}{N_s - 1}\, S_M^{-1}\, \Delta M\, \Delta M^T\, S_M^{-1} = \frac{1}{N_s - 1}\, S_M^{-1}\, \Delta M\, \left(S_M^{-1}\, \Delta M\right)^T\, . \tag{4.24}$$

As before, in Eq. 4.24, S_M^{-1} denotes the inverse of the standard deviation matrix, which is the diagonal matrix for which the jth entry of the diagonal is given by the inverse of the standard deviation of the jth parameter, i.e., the inverse of the standard deviation for the jth entry of the vector of reservoir model parameters $\boldsymbol{m}$. We do not assemble the matrix S_M^{-1} due to computational restrictions, let alone to compute the matrix product $S_M^{-1}\, \Delta M$. However, because the matrix S_M^{-1} is diagonal, one is able to compute the matrix product $S_M^{-1}\, \Delta M$ by simply multiplying the elements of the jth row of matrix ΔM, for $j = 1,\, 2,\, \cdots,\, N_m$, by the jth diagonal element of the matrix S_M^{-1}, which can be easily executed.

The aim in constructing the prior correlation matrix is to apply the PCA method to reduce the order of the problem. Although we cannot assemble the prior correlation matrix to apply PCA, the same result can be accomplished by applying SVD to the matrix

$S_M^{-1}\Delta M/\sqrt{N_s-1}$, see Shlens [110]. The left singular vectors of the matrix $S_M^{-1}\Delta M/\sqrt{N_s-1}$ are the same as the left singular vectors of the correlation matrix. Furthermore, the singular values of the matrix $S_M^{-1}\,\Delta M/\sqrt{N_s-1}$ represent the square root of the singular values of the correlation matrix, as can be shown by

$$\widetilde{C}_M \cong \frac{1}{N_s-1}\, S_M^{-1}\,\Delta M\,\left(S_M^{-1}\,\Delta M\right)^T = U_\Delta\, W_\Delta\, V_\Delta^T\, V_\Delta\, W_\Delta\, U_\Delta^T = U_\Delta\, W_\Delta^2\, U_\Delta^T\,, \qquad (4.25)$$

since $V_\Delta^T\, V_\Delta = I_{N_s}$. In Eq. 4.25, the matrices U_Δ, W_Δ and V_Δ represent, respectively, the matrix of left singular vectors, the matrix of singular values and the matrix of right singular vectors which are obtained by applying SVD to the matrix $S_M^{-1}\,\Delta M/\sqrt{N_s-1}$. Therefore, by applying SVD to the matrix $S_M^{-1}\,\Delta M/\sqrt{N_s-1}$, all elements that we need to apply the reduction order transformation of Eq. 3.47, see Section 3.3, become available, and we can apply our proposed methodology.

To apply the above described approach for this case, we sampled $N_s = 5,000$ models from the coarse prior pdf, which is exactly the same model as described in Section 4.4. Then, we convert the samples to the refined grid, and add white noise, to generate 5,000 samples for the large scale reservoir model. As commented earlier, this large scale model has dimension $N_m = 95,094$. Furthermore, for $j = 1, 2, \ldots, 47,547$, $m_j = \ln(k_j)$, where m_j represents the jth entry of the vector $\boldsymbol{m}$, and k_j represents the horizontal permeability of the jth gridblock. Similarly, for $j = 47,548, 47,549, \ldots, N_m$, $m_j = \ln(k_{z,j})$, where $k_{z,j}$ denotes the vertical permeability of the jth gridblock. Hence, as in the previous case, the horizontal and vertical log-permeability are the parameters considered for history matching.

With the 5,000 samples, we approximate the correlation matrix as depicted above, then we compute the matrix $S_M^{-1}\,\Delta M/\sqrt{N_s-1}$ and apply SVD. The results are presented in Fig. 4.56, in which we present the corresponding first 300 singular values of the prior correlation matrix. For a $p_{N_v} = 0.9679$, see the discussion preceding Eq. 3.42, we need to keep only 171 singular values, i.e., $N_v = 171$ for this case, see the discussion preceding Eq. 3.40. The vertical dashed line in Fig. 4.56 marks the position of the 171st singular value.

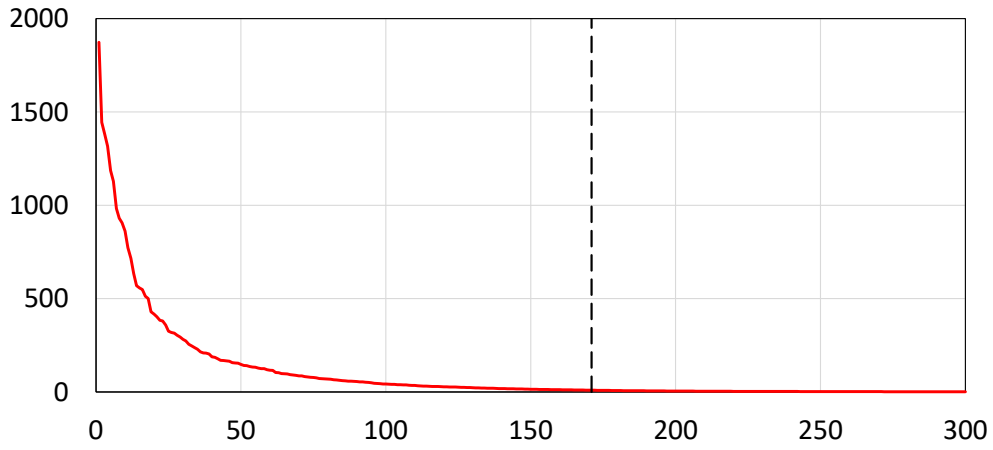


Figure 4.56: The SVD results of the prior correlation matrix for the large scale case: the red curve represents the first 300 singular values, the dashed vertical line represents the position of the 171st singular value.

Notice that we reduce the original dimension of the problem, i.e., $N_m = 95,094$, to only 171.

For this large scale problem, we generated a new true horizontal and vertical permeability fields. The new true models was sampled from the coarse model prior pdf, then converted to the refined grid. The resulting log-permeability fields are presented in Figs. 4.57 and 4.58

For this large scale problem, we use a total of eleven wells, six production wells and five injection wells. As before, the producers are denoted as PRO-01, PRO-04, PRO-05, PRO-11, PRO-12, and PRO-15. The injectors are denoted as INJ-01, INJ-02, INJ-03, INJ-04, and INJ-05. The well gridblock completion information is presented in Table 4.6. Note there are now 15 reservoir simulation model layers. The producers PRO-01, PRO-04, PRO-05, PRO-11, and PRO-12 are operated at a constant oil flow rate of 628.98 STB/day, which is equivalent to 100 m^3/day. The producer PRO-15 is operated at a constant oil flow rate of 314.49 STB/day, which is equivalent to 50 m^3/day. All injectors are operated at a constant water flow rate of 691.88 STB/day, which is equivalent to 110 sm^3/day.

For the large scale case, the data considered for history matching are the bottom-hole pressure, the gas production rate and the water production rate for each one of the six producers, and the bottom-hole pressure for each one of the five injectors. As in the previous case, the data are acquired every 30 days. A total of $96 \times 30 = 2,880$ days (almost

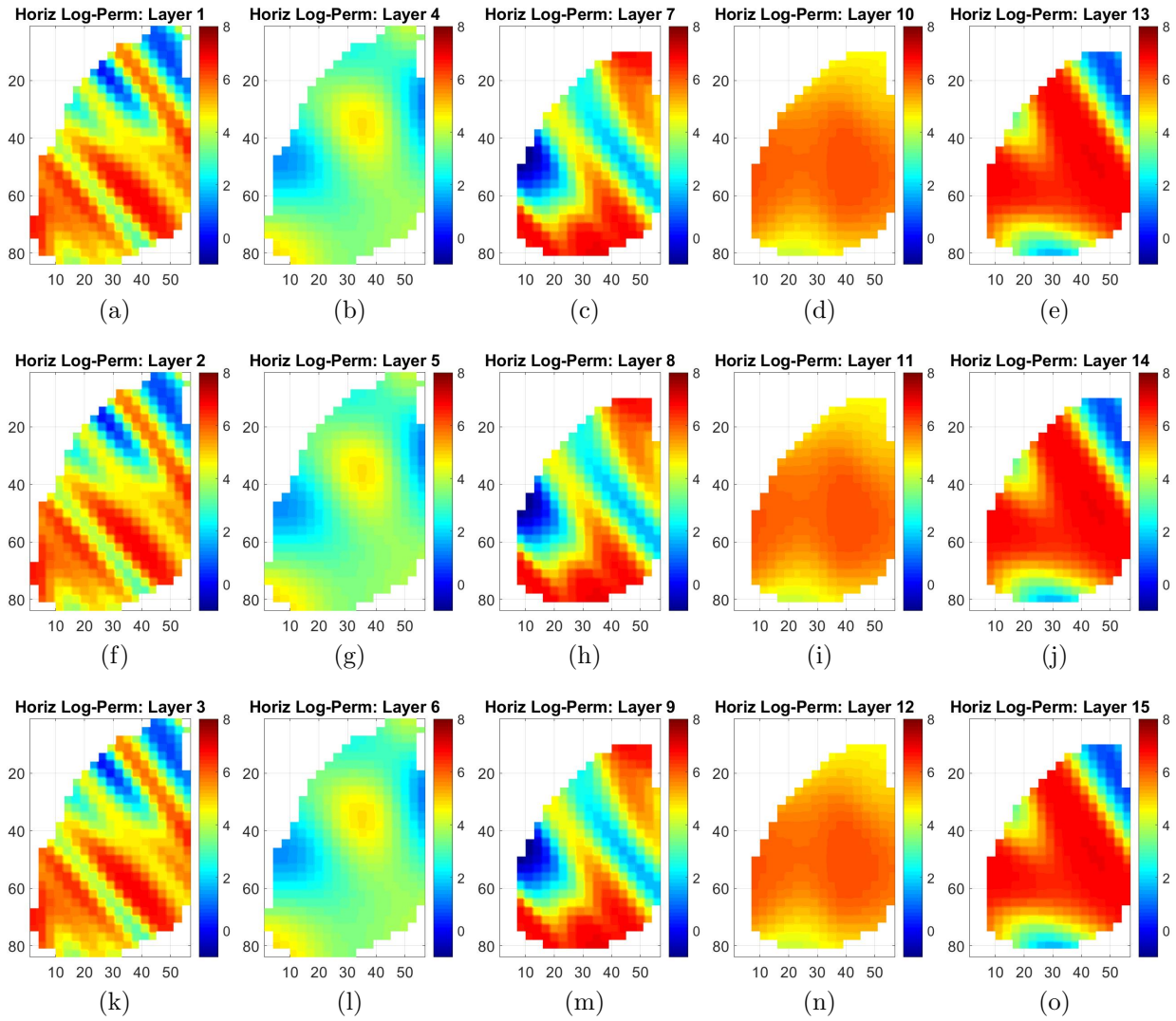


Figure 4.57: True horizontal log-permeability field for the large scale case.

8 years) history matching period is considered, which results in $N_{d_h} = 23 \times 96 = 2,208$ observed production data, i.e., 3×6 data for producers and 5×1 data for injectors each 30 days. The history matching period is followed by $48 \times 30 = 1,440$ days, or almost 4 years, of forecast period. Therefore, $N_d = 23 \times (96 + 48) = 3,312$. As a consequence, to add a point to the training set we run the reservoir simulator for $(96 + 48) \times 30 = 4,320$ days, or almost 12 years. For this case, the standard deviation of measurement errors for the bottom-hole pressure is set equal to 5 bars for all six producers and all five injectors. Meanwhile, the standard deviation of measurement errors for gas production rate is set equal to 100 sm^3/day for all six producers. Finally, the standard deviation of measurement errors

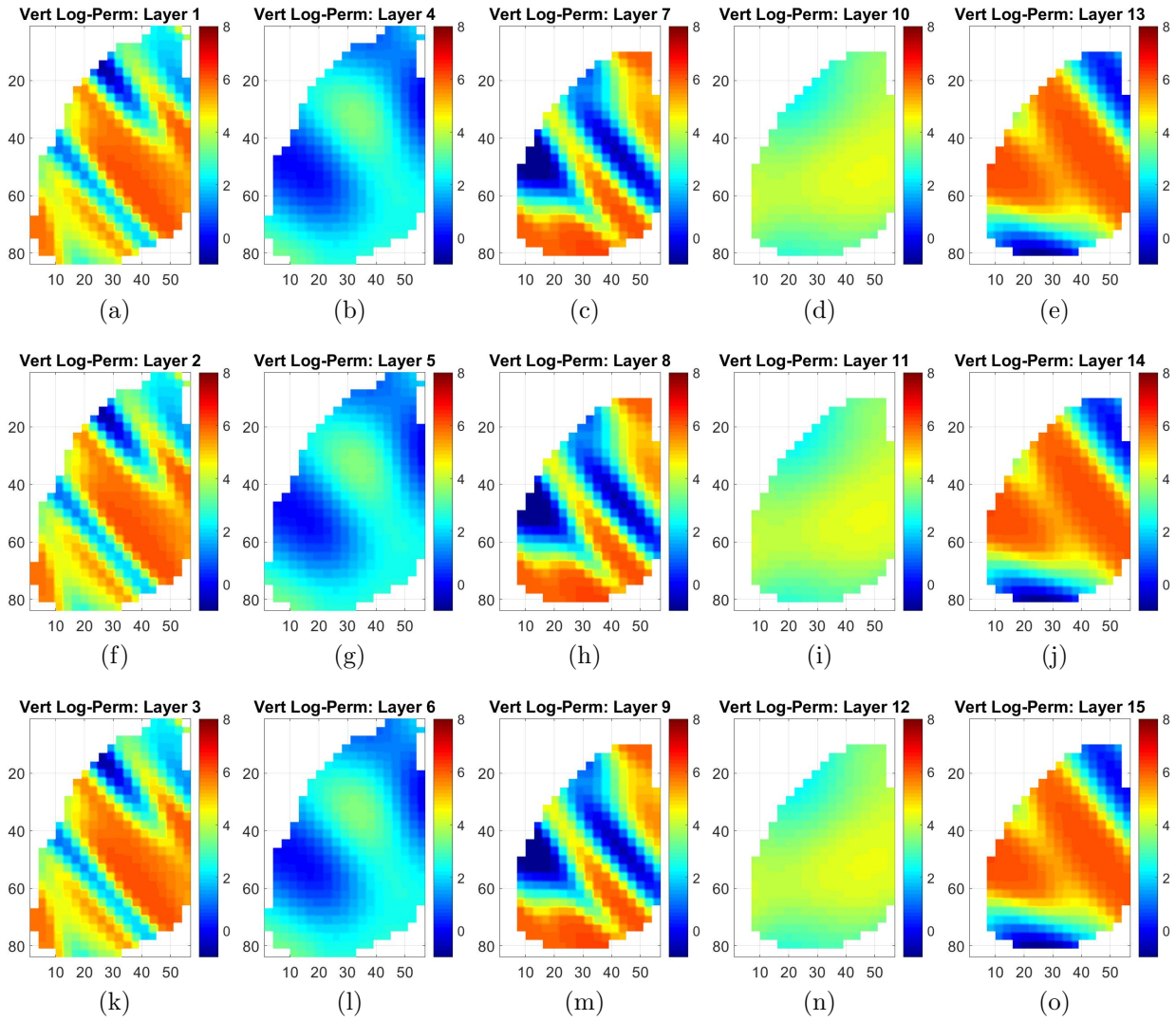


Figure 4.58: True vertical log-permeability field for the large scale case.

for water production rate is set equal to 0.50 sm^3/day for all six producers. With these data, one is able to construct the measurement error covariance matrix.

We consistently chose $N_e = 250$ and conducted 250 minimization problems as for the previous cases. It is possible that for a more complex problem, such as this large scale reservoir model, one should increase the value of N_e to achieve a better characterization of the posterior pdf and find as many modes as possible. Nevertheless, each reservoir simulation for this problem takes 5 to 8 minutes, so increasing the value of N_e is not computational feasible for us. In fact, to be able to complete the minimization loop in an affordable time, we had to reduce this value to $N_e = 100$ for this large scale problem. Performing only 100 minimization

Well	gridblock Completion		
	$x-$dir	$y-$dir	$z-$dir
PRO-01	29	65	10, 11, 12, 13, 14 and 15
PRO-04	26	50	10, 11, 12, 13, 14 and 15
PRO-05	50	32	7, 8, 9, 10, 11 and 12
PRO-11	32	71	7, 8, 9, 10, 11 and 12
PRO-12	44	35	10, 11, and 12
PRO-15	50	65	10, 11, 12, 13, 14 and 15
INJ-01	29	20	10, 11, 12, 13, 14 and 15
INJ-02	14	38	10, 11, 12, 13, 14 and 15
INJ-03	8	53	10, 11, 12, 13, 14 and 15
INJ-04	8	71	10, 11, 12, 13, 14 and 15
INJ-05	20	80	10, 11, 12, 13, 14 and 15

Table 4.6: Producer and injectors wells gridblock completion for the large scale case.

problems may not be enough to approximately characterize the posterior pdf when searching for modes, and it probably will affect the quality of the resulting uncertainty quantification. However, the aim here is to evaluate the performance of the proposed methodology for a large scale problem, which had to be done under a computational limitation. Nevertheless, the results will show that the proposed methodology, using the LS-SVR proxy, was able to find modes of the posterior pdf, which ultimately results in a reasonable uncertainty quantification, although some bias is observed, most probably due to a lack of representation of the modes of the posterior pdf. The $N_e = 100$ initial models are generated first from sampling the coarse prior pdf using LHS [74]. An additional of $N_a = 400$ models are also generated in the same way to compose an initial training set with $N_t = 500$ training examples, which is small compared with the size of the initial training set we adopted for the other applications. As in the previous case, we opt to condition those $N_t = 500$ models for log-permeability hard data. A new hard data was randomly generated from the new true log-permeability field by adding white noise with standard deviation equal to 20% of the actual gridblock permeability value. Due to computational limitations, we conditioned the coarse models to the new hard data, then we converted them to the refined grid. Similarly to the previous case, all vertical gridblocks penetrated by a producer well were consider as the location for which the new hard data was generated. The procedure to condition the

models to hard data is the same depicted in the previous case.

To construct the initial training set, we run the reservoir simulator for each one of the conditioned to hard data initial models, $\boldsymbol{m}_{k,\mathrm{hd}}$, $k = 1, \ldots, N_t = 500$. Then, as before, the conditioned initial models are transformed to their reduced order counterparts, see Eq. 3.47, using

$$\boldsymbol{z}_k = W_{N_v}^{-1/2}\, U_{N_v}^T\, S_M^{-1}\, (\boldsymbol{m}_{k,\mathrm{hd}} - \boldsymbol{m}_{\mathrm{pr}})\,, \qquad \text{for} \quad k = 1,\, 2,\, \ldots,\, N_t\,. \tag{4.26}$$

In Eq. 4.26, $N_v = 171$, as discussed earlier. The initial training set is then constructed using the vectors $\boldsymbol{z}_k$, for $k = 1, \ldots, N_t = 500$, and the associated reservoir predictions. Again it is worthwhile to mention that we only condition the initial models to the hard data. Subsequent models added to the training set are from the minimization steps and are not conditioned to the hard data.

Following the proposed methodology, we conduct the minimization problems in the reduced order space of the vector $\boldsymbol{z}$. When we need to add a new point to the training set, we convert the corresponding $\boldsymbol{z}$ to its respective $\boldsymbol{m}$, using Eq. 3.49, then we run the reservoir simulator. For this large scale problem, we adopt a minimum distance to update the training set of $d_{\min} = 1.0\mathrm{E}-4$. The trust-region parameters are set to $\delta_e^{(1)} = 0.2$, for $e = 1, \ldots, N_e$, $\delta_{\min} = 2.0\mathrm{E}-4$ and $\delta_{\max} = 1.0$. Note that in this example, we adopt less restrictive values than used before in order to accelerate the convergence of the minimization problems, since each iteration takes a significantly time due to the considerable time required for each reservoir simulation run. The convergence criteria are set to $\epsilon_1 = \epsilon_2 = 1.0\mathrm{E}-4$, which is also less restrictive than used in the previous case. Finally, $p_{\mathrm{conv}} = 0.90$, so a minimum of 90% of the minimization problems have to converge. As before, numerical experiments using the $N_e = 100$ selected models as the test set and using the $N_a = 400$ models to construct a training set are conducted to determine the LS-SVR proxy parameters, which results in a $\gamma = 800$ and $\sigma = 2.6$, corresponding to a $c_\sigma = 0.20$, see Eq. 3.9. During the minimization loop, we set $N_{\mathrm{cut}} = 3,500$ and $O_{N,\mathrm{cut}} = 80.0$, which means that when the training set becomes larger than 3,500 training examples, then at each iteration of the minimization

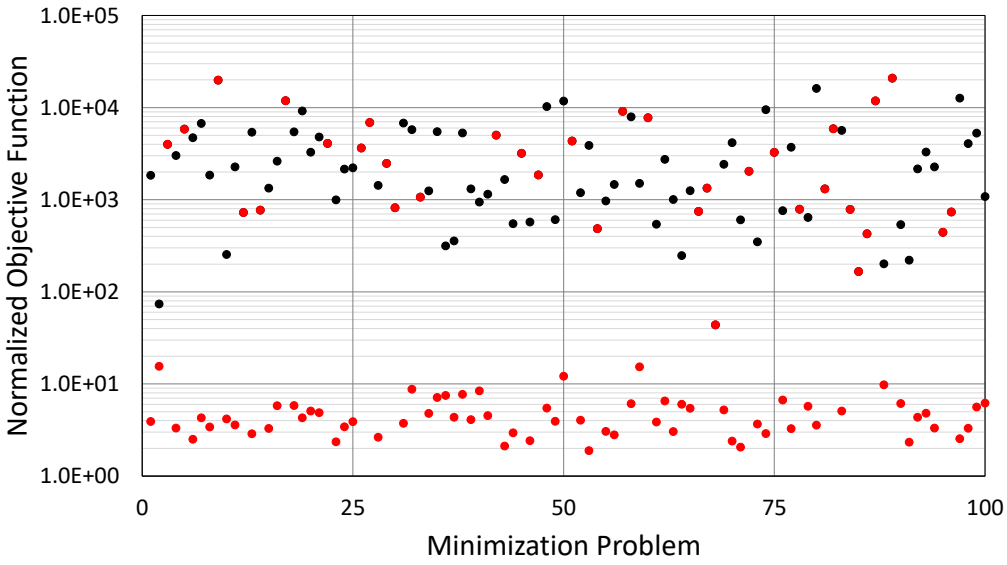


Figure 4.59: Minimization loop results for the large scale case: prior models (black dots), converged models (red dots).

loop, we remove from the training set the training examples which give normalized objective function values greater than 80.0, see the discussion preceding Eq. 3.27.

In Fig. 4.59 we present the minimization loop results. As in the previous cases, the final values of normalized objective function obtained after convergence are about four orders of magnitude lower than the corresponding values for the prior models, even though we considerably reduced the dimension of the problem from 95,094 to only 171. Although the LS-SVR proxy model is trained for the reduced order vector, which has dimension 171, the minimization results corroborates that the LS-SVR proxy gradient is indeed a good alternative to the adjoint solution.

However, Eq. 3.27, which gives the approximate upper bound for the value of the normalized objective function at the minima, provides an upper bound of 1.15 for this case, and the minimum value obtained in the minimization loop is 1.89. Again, the performance for the LS-SVR gradient for this case is slightly inferior to the performance observed in the one-dimensional case. Nevertheless, 51 models give normalized objective function values less than 6.0, which, considering that a small initial training set was adopted due computational limitations, indicates a reasonably good performance of the proposed methodology.

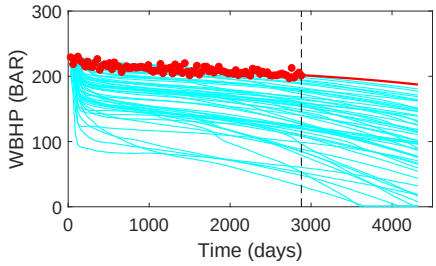
At the end of the minimization loop, a total of 2,852 reservoir simulation runs were

required in order for 90% of the 100 minimization problems to converge. This resulted in a training set with 2,681 training examples. Using a value of $O_{N\max} = 20.0$, we reduced the size of the training set to a total of 1,636 training examples. To further reduce the training set size, we applied the seed procedure described in Subsection 3.2.1 of Chapter 3. For this case, we selected the found modes which presented normalized objective function values less than 10.0, which represents 63 modes, to start the seed for a new training set. Then, we compare the models from the training set with the seed models using a normalized distance of $1.0\text{E}-3$, for the cases were the distance for the models in the seed is larger than the adopted distance, the model is added to the seed. At the end, 1,441 training examples were selected to form the final training set. Then, we use this final training set to train the final LS-SVR proxy model.

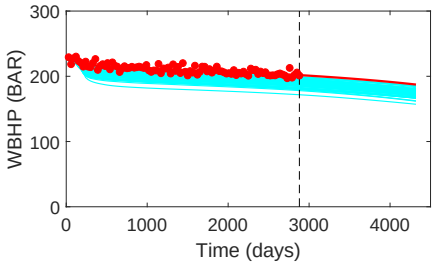
From the 100 minimization problems, 51 converge to modes that give a normalized objective function value less than 6.0 and are used to construct the GMM approximation. In Figs. 4.60 through 4.67 we present the history matching results for those 51 selected modes. The original prior predictions for the 51 models are compared to the predictions after history matching. As one can see, even though we conducted the minimization loop using a relatively small initial training set, the overall history matching performance for these 51 selected model are quite good. In all figures, the solid thick red curve represents the true value, the red dots represent the observed data, and the cyan curves represents the corresponding models predictions.

In Figs. 4.68 through 4.103 we compare the prior horizontal and vertical log-permeability fields with the corresponding conditioned models after history matching for the first six selected modes. As before, even though we conducted the minimization problems using the vector $\boldsymbol{z}$, which has dimension 171, and the LS-SVR proxy models was trained using the vector $\boldsymbol{z}$, the overall minimization results show permeability fields which are consistent with the prior models, i.e., again the minimization process preserved the geological information.

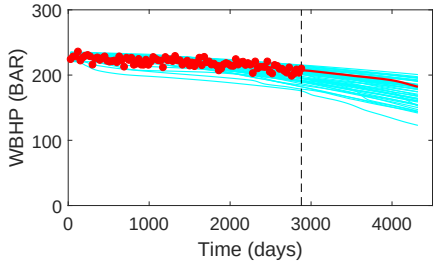
As before, the selected 51 modes are clustered into $N_c = 25$ clusters using the k-medoids clustering algorithm [63]. Then, the clusters are used to build a GMM approxi-



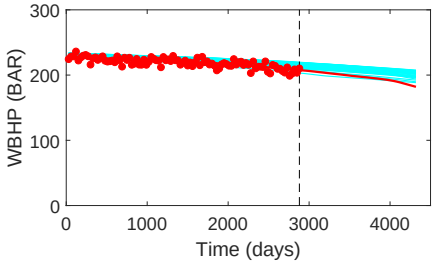
(a) BHP PRO-01: before history matching.



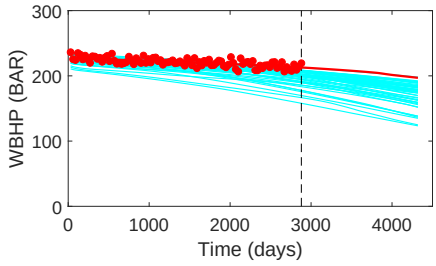
(b) BHP PRO-01: after history matching.



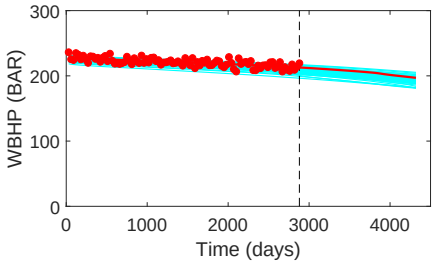
(c) BHP PRO-04: before history matching.



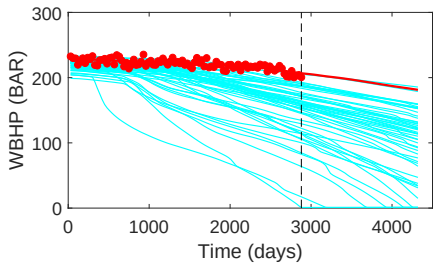
(d) BHP PRO-04: after history matching.



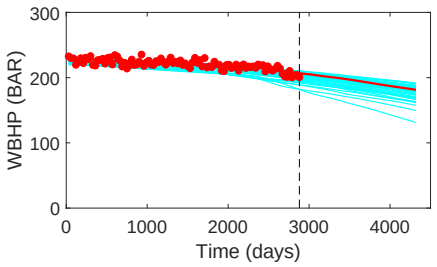
(e) BHP PRO-05: before history matching.



(f) BHP PRO-05: after history matching.

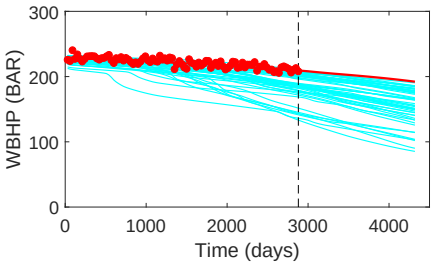


(g) BHP PRO-11: before history matching.

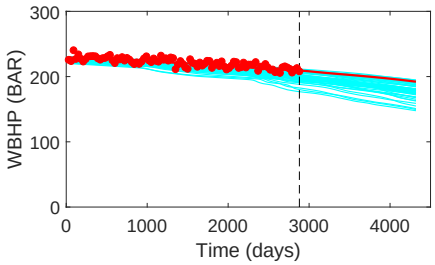


(h) BHP PRO-11: after history matching.

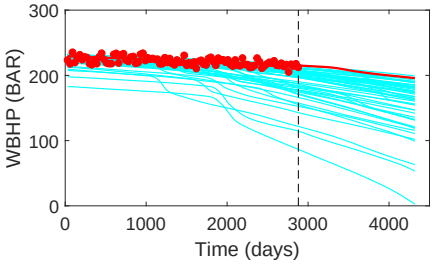
Figure 4.60: History matching results for the large scale case: well bottom-hole pressure at producers PRO-01, PRO-04, PRO-05, and PRO-11. A comparison between the original prior models predictions (left column) with the predictions after the history matching (right column) is presented. In all figures, we show the corresponding true value (solid thick red curve), the observed data (red dots), and the models predictions (cyan curves).



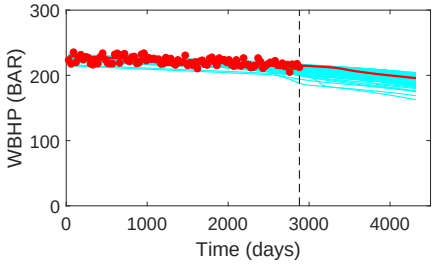
(a) BHP PRO-12: before history matching.



(b) BHP PRO-12: after history matching.

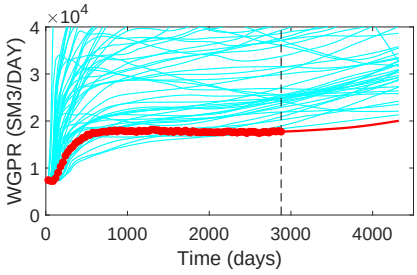


(c) BHP PRO-15: before history matching.

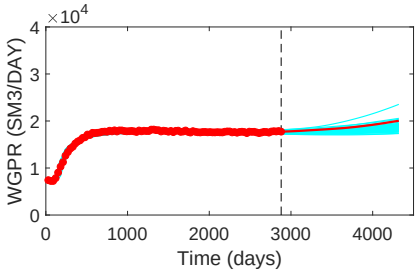


(d) BHP PRO-15: after history matching.

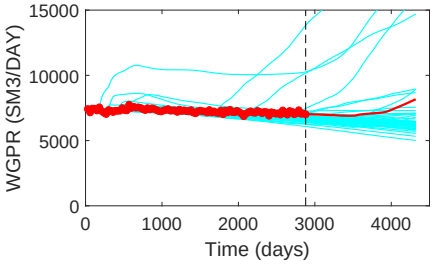
Figure 4.61: Same as in Fig. 4.60 for producers PRO-12 and PRO-15.



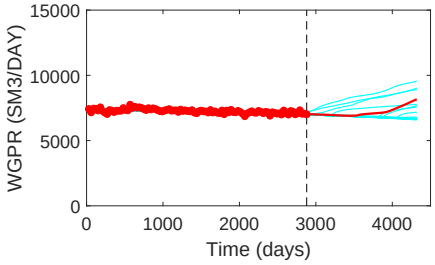
(a) GPR PRO-01: before history matching.



(b) GPR PRO-01: after history matching.

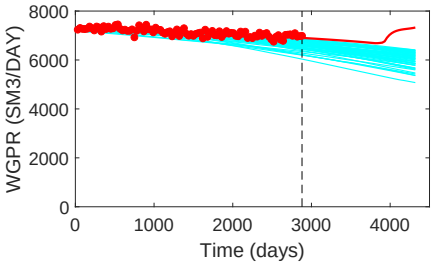


(c) GPR PRO-04: before history matching.

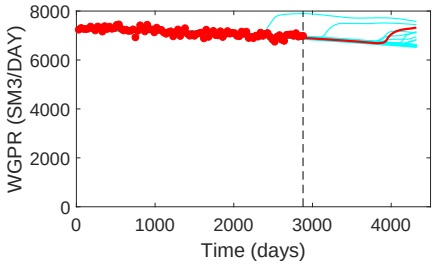


(d) GPR PRO-04: after history matching.

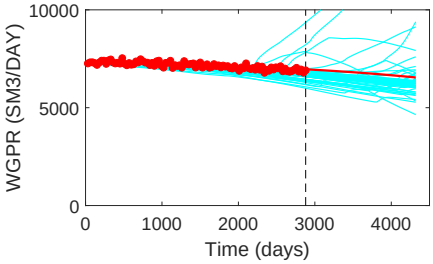
Figure 4.62: History matching results for the large scale case: gas production rate at producers PRO-01 and PRO-04. A comparison between the original prior models predictions (left column) with the predictions after the history matching (right column) is presented. In all figures, we show the corresponding true value (solid thick red curve), the observed data (red dots), and the models predictions (cyan curves).



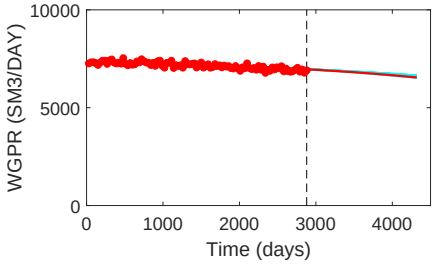
(a) GPR PRO-05: before history matching.



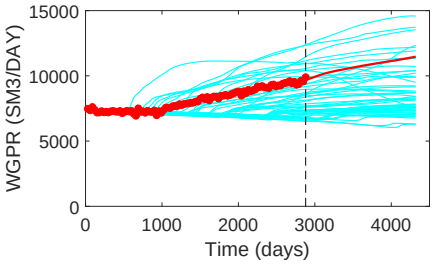
(b) GPR PRO-05: after history matching.



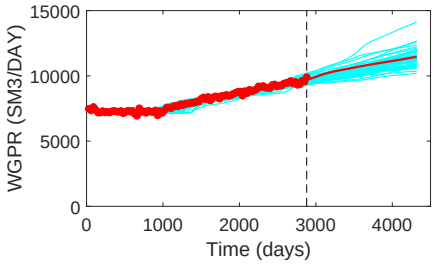
(c) GPR PRO-11: before history matching.



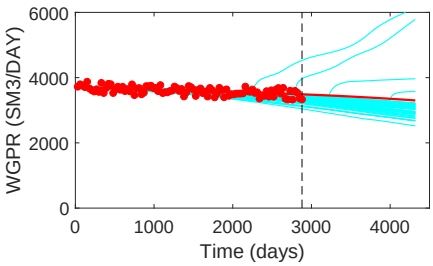
(d) GPR PRO-11: after history matching.



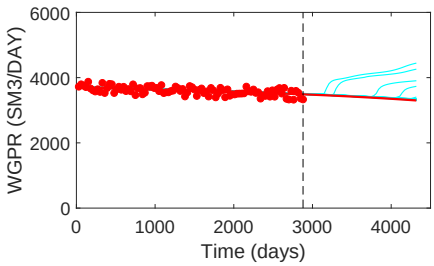
(e) GPR PRO-12: before history matching.



(f) GPR PRO-12: after history matching.

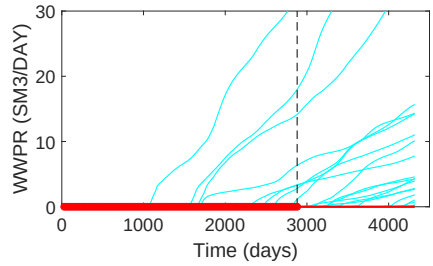


(g) GPR PRO-15: before history matching.

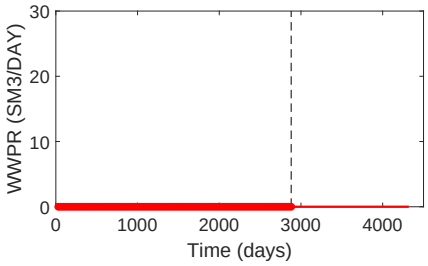


(h) GPR PRO-15: after history matching.

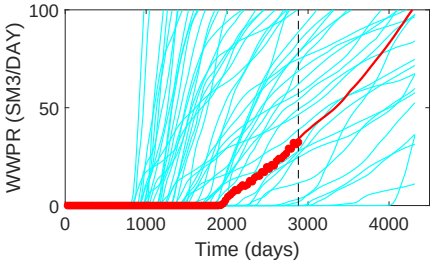
Figure 4.63: History matching results for the large scale case: gas production rate at producers PRO-05, PRO-11, PRO-12, and PRO-15. A comparison between the original prior models predictions (left column) with the predictions after the history matching (right column) is presented. In all figures, we show the corresponding true value (solid thick red curve), the observed data (red dots), and the models predictions (cyan curves).



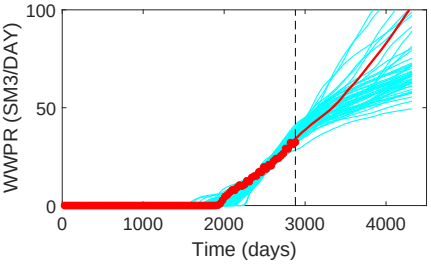
(a) WPR PRO-01: before history matching.



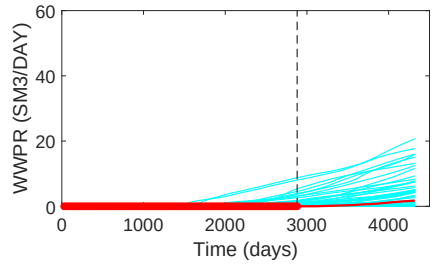
(b) WPR PRO-01: after history matching.



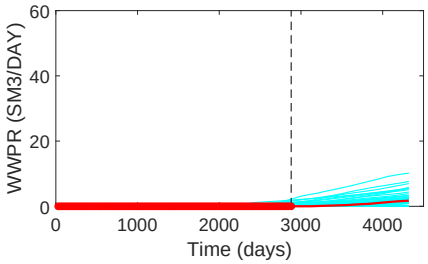
(c) WPR PRO-04: before history matching.



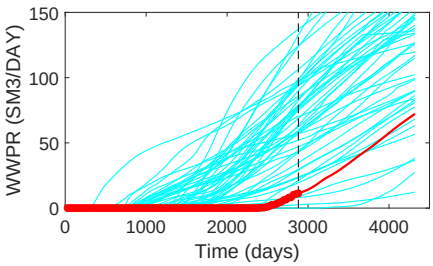
(d) WPR PRO-04: after history matching.



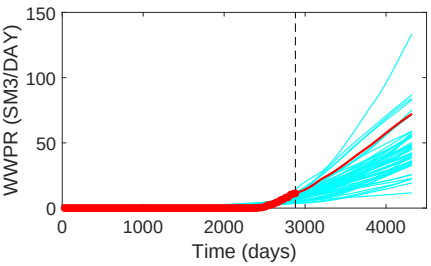
(e) WPR PRO-05: before history matching.



(f) WPR PRO-05: after history matching.

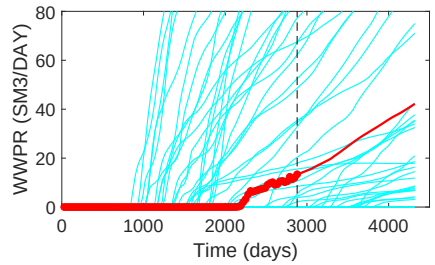


(g) WPR PRO-11: before history matching.

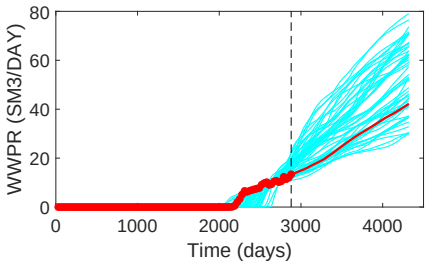


(h) WPR PRO-11: after history matching.

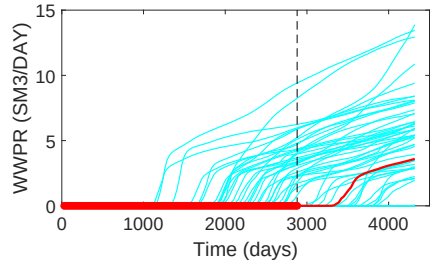
Figure 4.64: History matching results for the large scale case: water production rate at producers PRO-01, PRO-04, PRO-05, and PRO-11. A comparison between the original prior models predictions (left column) with the predictions after the history matching (right column) is presented. In all figures, we show the corresponding true value (solid thick red curve), the observed data (red dots), and the models predictions (cyan curves).



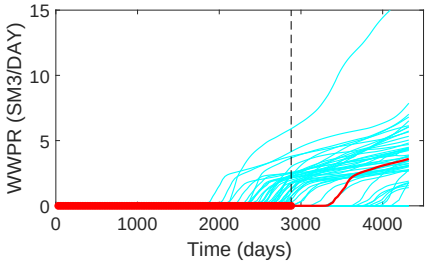
(a) WPR PRO-12: before history matching.



(b) WPR PRO-12: after history matching.

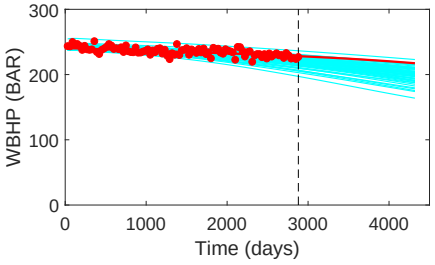


(c) WPR PRO-15: before history matching.

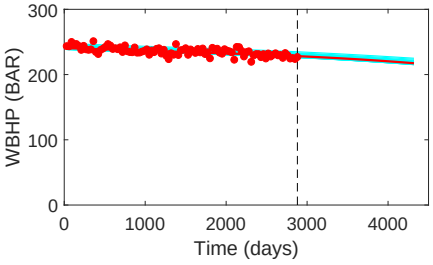


(d) WPR PRO-15: after history matching.

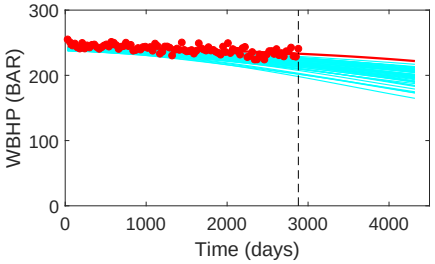
Figure 4.65: Same as in Fig. 4.64 for producers PRO-12 and PRO-15.



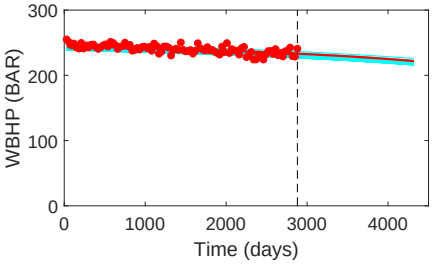
(a) BHP INJ-01: before history matching.



(b) BHP INJ-01: after history matching.

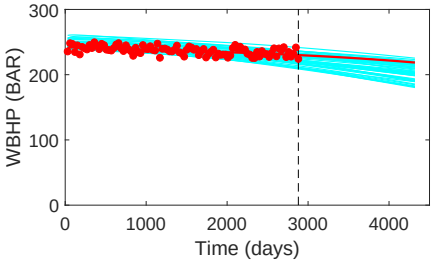


(c) BHP INJ-02: before history matching.

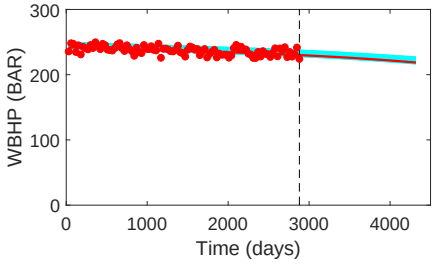


(d) BHP INJ-02: after history matching.

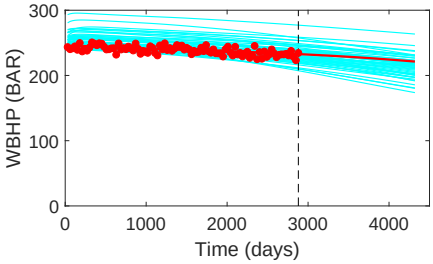
Figure 4.66: History matching results for the large scale case: well bottom-hole pressure at injectors INJ-01 and INJ-02. A comparison between the original prior models predictions (left column) with the predictions after the history matching (right column) is presented. In all figures, we show the corresponding true value (solid thick red curve), the observed data (red dots), and the models predictions (cyan curves).



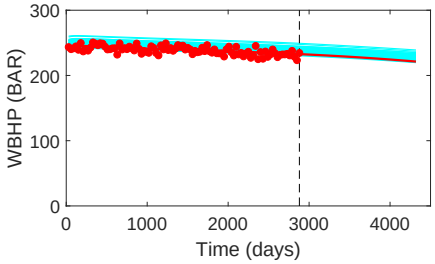
(a) BHP INJ-03: before history matching.



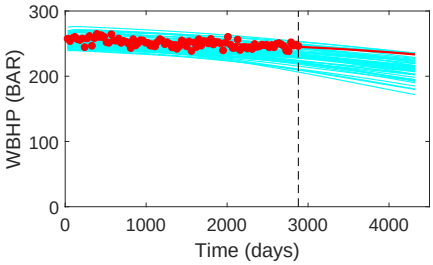
(b) BHP INJ-03: after history matching.



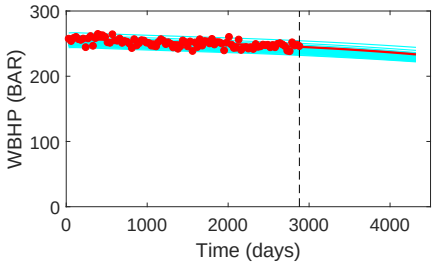
(c) BHP INJ-04: before history matching.



(d) BHP INJ-04: after history matching.



(e) BHP INJ-05: before history matching.



(f) BHP INJ-05: after history matching.

Figure 4.67: History matching results for the large scale case: well bottom-hole pressure at injectors INJ-03, INJ-04 and INJ-05. A comparison between the original prior models predictions (left column) with the predictions after the history matching (right column) is presented. In all figures, we show the corresponding true value (solid thick red curve), the observed data (red dots), and the models predictions (cyan curves).

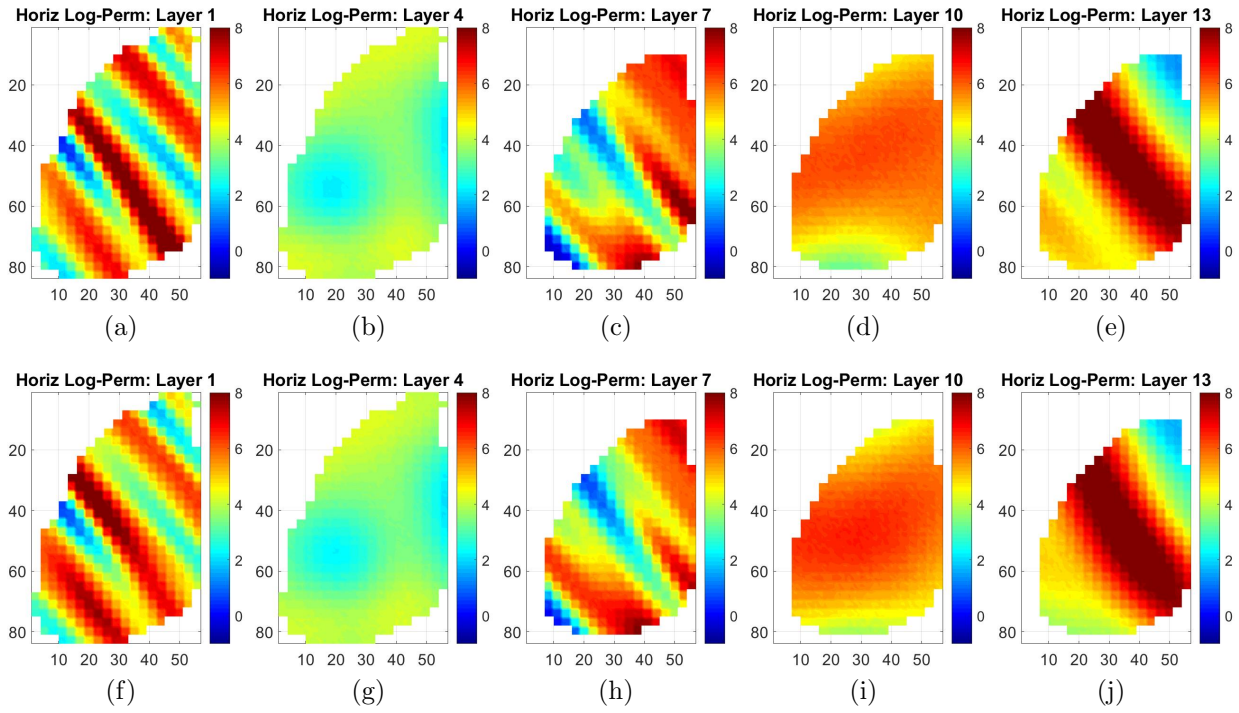


Figure 4.68: Horizontal log-permeability fields for the large scale case comparing the prior model (top row, layers 1, 4, 7, 10 and 13) with the history matched model (bottom row, layers 1, 4, 7, 10 and 13) for the converged model 1.

mation for the posterior pdf. Again, the weights in the GMM approximation are all set to $w_\ell = 1/25 = 0.04$ for $\ell = 1, \ldots, N_c$. Using this GMM approximation as the proposal distribution, we run five chains with a total of 100,000 states each. As before, we first run short chains of length 2,000 states to tune the parameter μ_γ. For this case, we selected a parameter $\mu_\gamma = 40.0$, leading to a learning rate of $\gamma_i = 1.37\text{E} - 3$, which results in an acceptance rate of 32.1%. The convergence of the five chains based on the MPSRF method [14] is presented in Fig. 4.104. As one can see, the MPSRF reaches unity after 25,000 states are generated. Therefore, from state 25,000 onward, MPSRF indicates that the chains have converged to the posterior pdf, which again is a fast convergence.

Figs. 4.105 and 4.106 present, respectively, the resulting posterior mean for both horizontal and vertical gridblock log-permeability fields. Comparing the posterior mean with the corresponding true log-permeability fields of Figs. 4.57 and 4.58, one can see that some features of the true log-permeability field are represented in the posterior mean, even

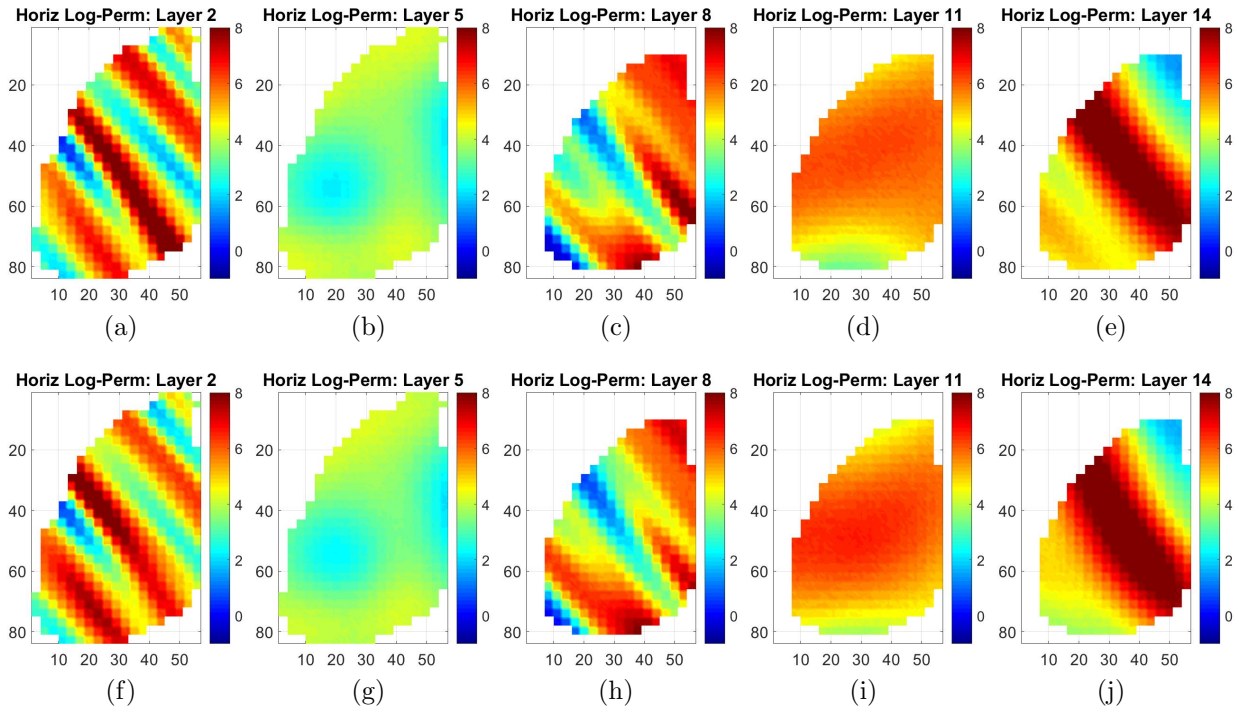


Figure 4.69: Horizontal log-permeability fields for the large scale case comparing the prior model (top row, layers 2, 5, 8, 11 and 14) with the history matched model (bottom row, layers 2, 5, 8, 11 and 14) for the converged model 1.

though as in the previous case, the posterior mean fields are smoother than the truth and individual posterior realizations.

The uncertainty quantification results are presented in Figs. 4.107 through 4.110. To generate the results presented in Figs. 4.53 through 4.55, we reconstruct the marginal distributions using all states after state 40,000 for each one of the five chains, which results in a total of 300,000 states. The color code in all these figures is the same as described in Fig. 4.5, i.e., the solid thick black curve represents the mean of the marginal distribution, the two blue dashed curves represents the 25% (P25) and 75% (P75) percentiles, while the two solid thin black curves represent the 5% (P5) and 95% (P95) percentiles, finally, the solid thick red curve represents the predictions using the true model and the red dots represent the observed data. The marginal distribution for well bottom hole pressure (WBHP) at the producers is presented in Fig. 4.107. The marginal distribution for well gas production rate (WGPR) at the producers is presented in Fig. 4.108. The marginal distribution for well water

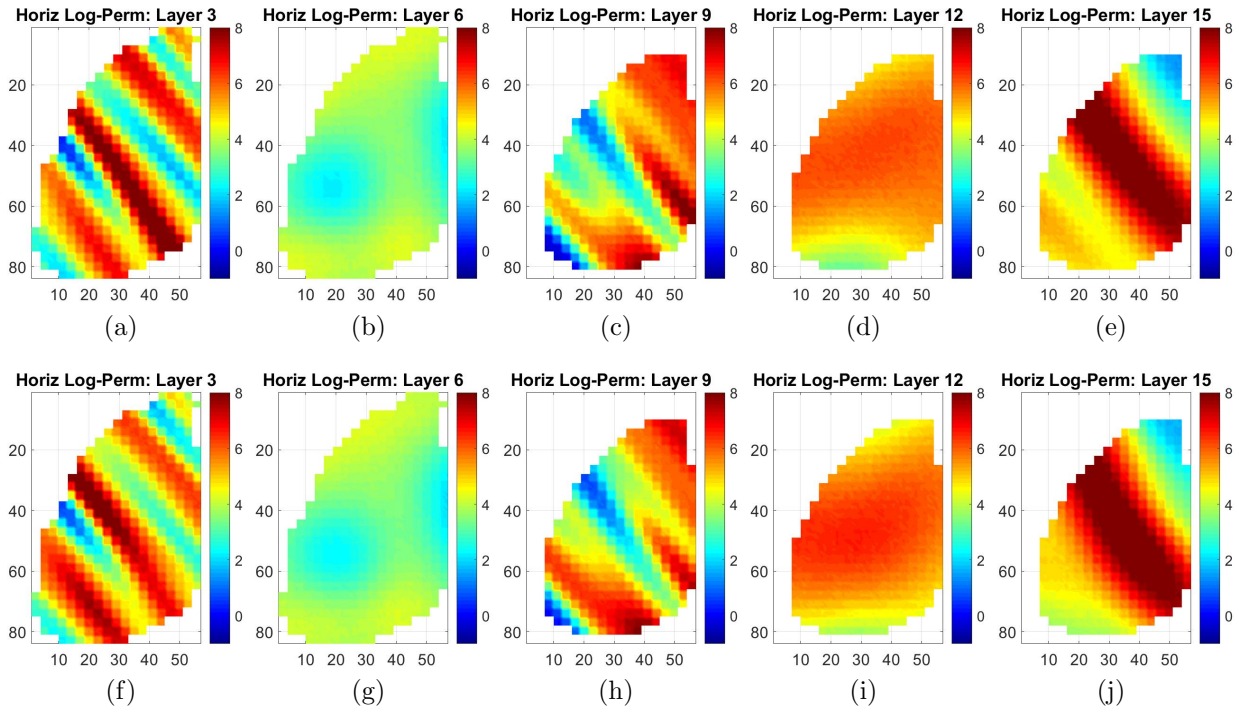


Figure 4.70: Horizontal log-permeability fields for the large scale case comparing the prior model (top row, layers 3, 6, 9, 12 and 15) with the history matched model (bottom row, layers 3, 6, 9, 12 and 15) for the converged model 1.

production rate (WWPR) at the producers is presented in Fig. 4.109. Finally, the marginal distribution for well bottom hole pressure at the injectors is presented in Fig. 4.110. Except for the well bottom hole pressure of PRO-01, PRO-04 and INJ-02, which present some bias, the overall performance of the uncertainty quantification seems remarkably good. Again, no base case is available for comparison, and we do believe that for a real case one should consider more than 100 minimization problems and use a larger initial training set. Nevertheless, the history matching results which can be achieved using the proposed methodology, and the associated low computational cost, seems promising for practical applications.

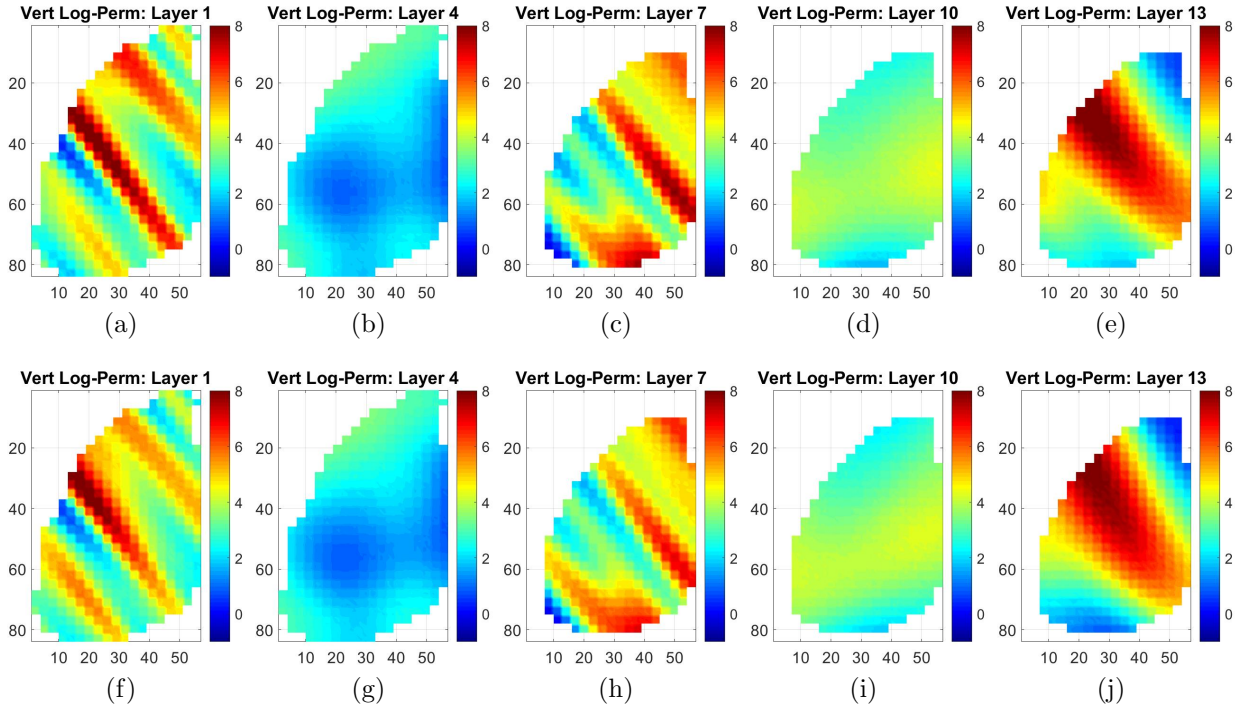


Figure 4.71: Vertical log-permeability fields for the large scale case comparing the prior model (top row, layers 1, 4, 7, 10 and 13) with the history matched model (bottom row, layers 1, 4, 7, 10 and 13) for the converged model 1.

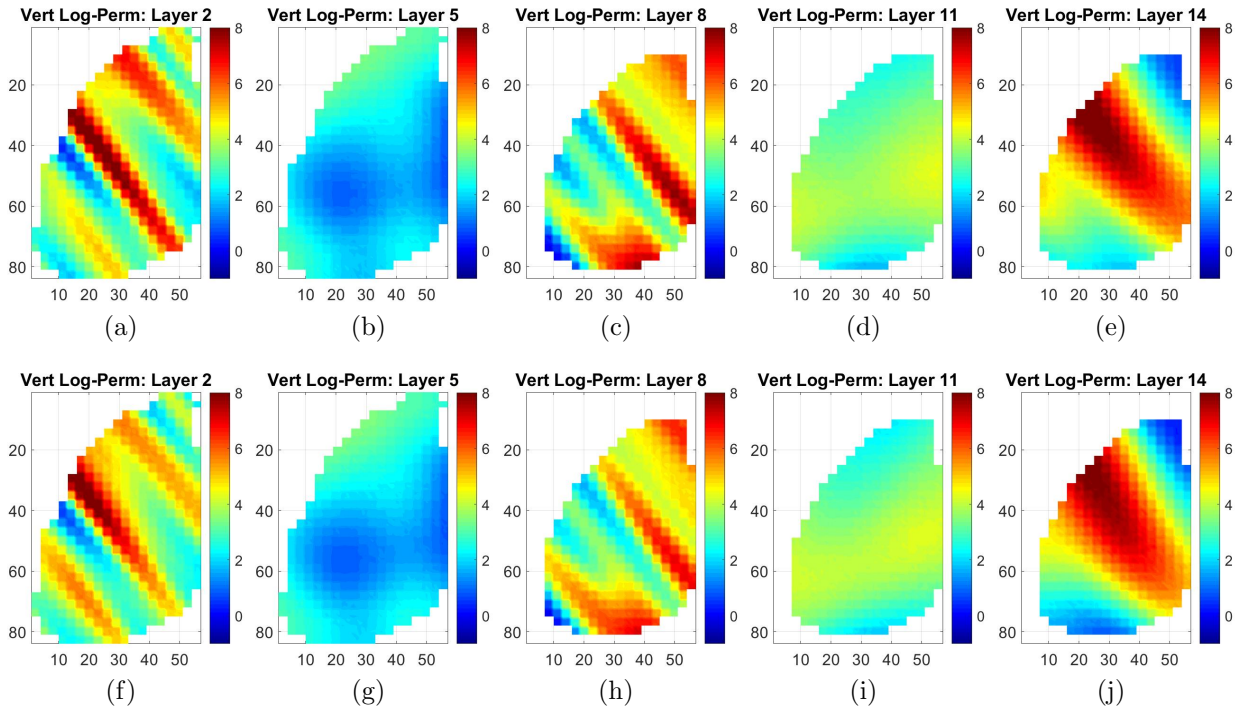


Figure 4.72: Vertical log-permeability fields for the large scale case comparing the prior model (top row, layers 2, 5, 8, 11 and 14) with the history matched model (bottom row, layers 2, 5, 8, 11 and 14) for the converged model 1.

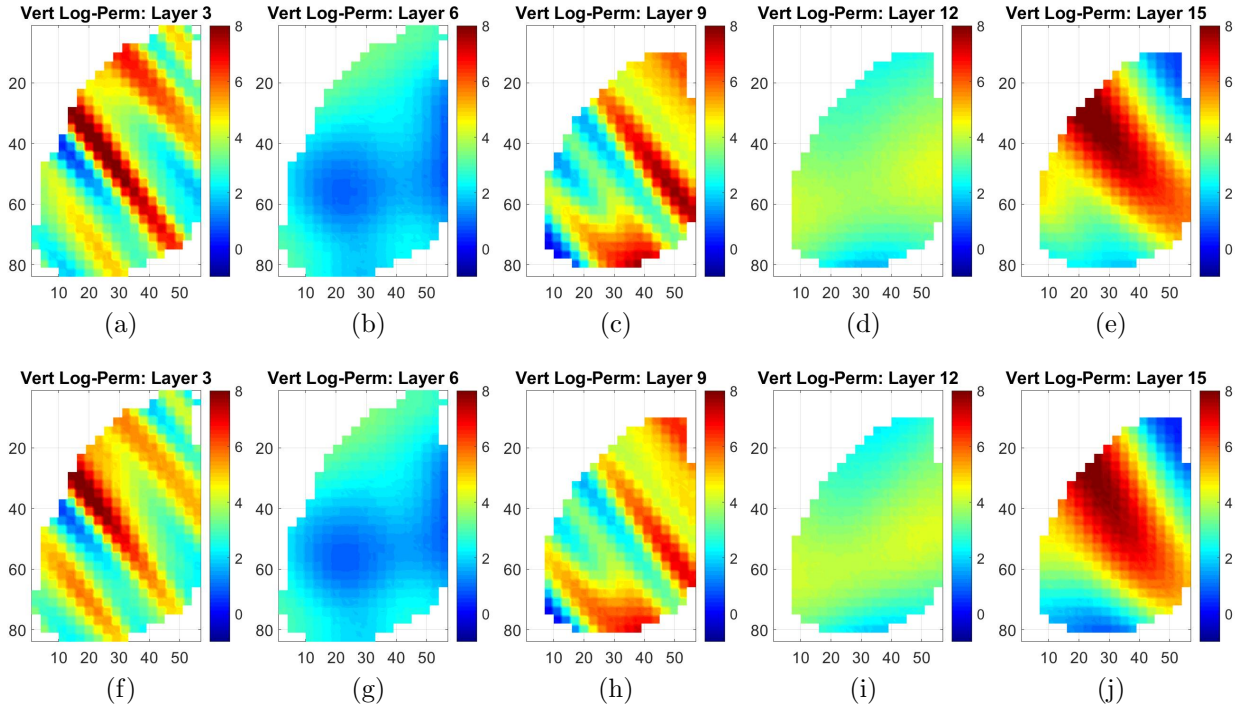


Figure 4.73: Vertical log-permeability fields for the large scale case comparing the prior model (top row, layers 3, 6, 9, 12 and 15) with the history matched model (bottom row, layers 3, 6, 9, 12 and 15) for the converged model 1.

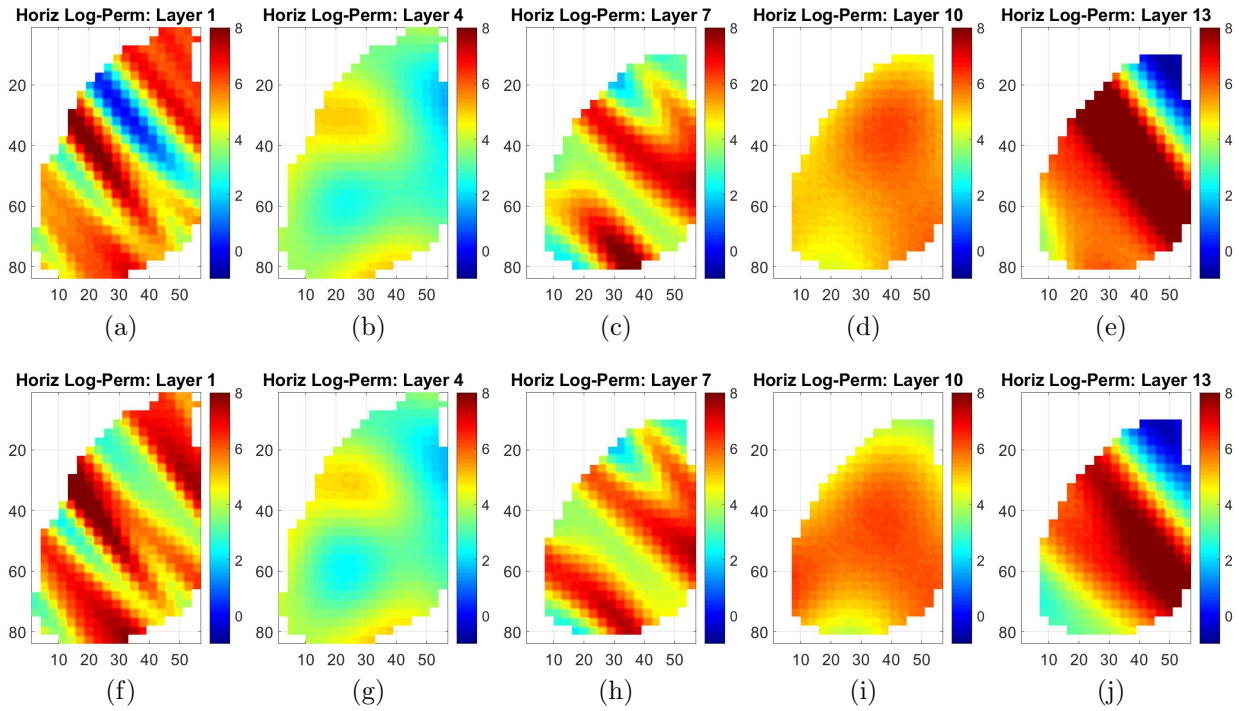


Figure 4.74: Horizontal log-permeability fields for the large scale case comparing the prior model (top row, layers 1, 4, 7, 10 and 13) with the history matched model (bottom row, layers 1, 4, 7, 10 and 13) for the converged model 2.

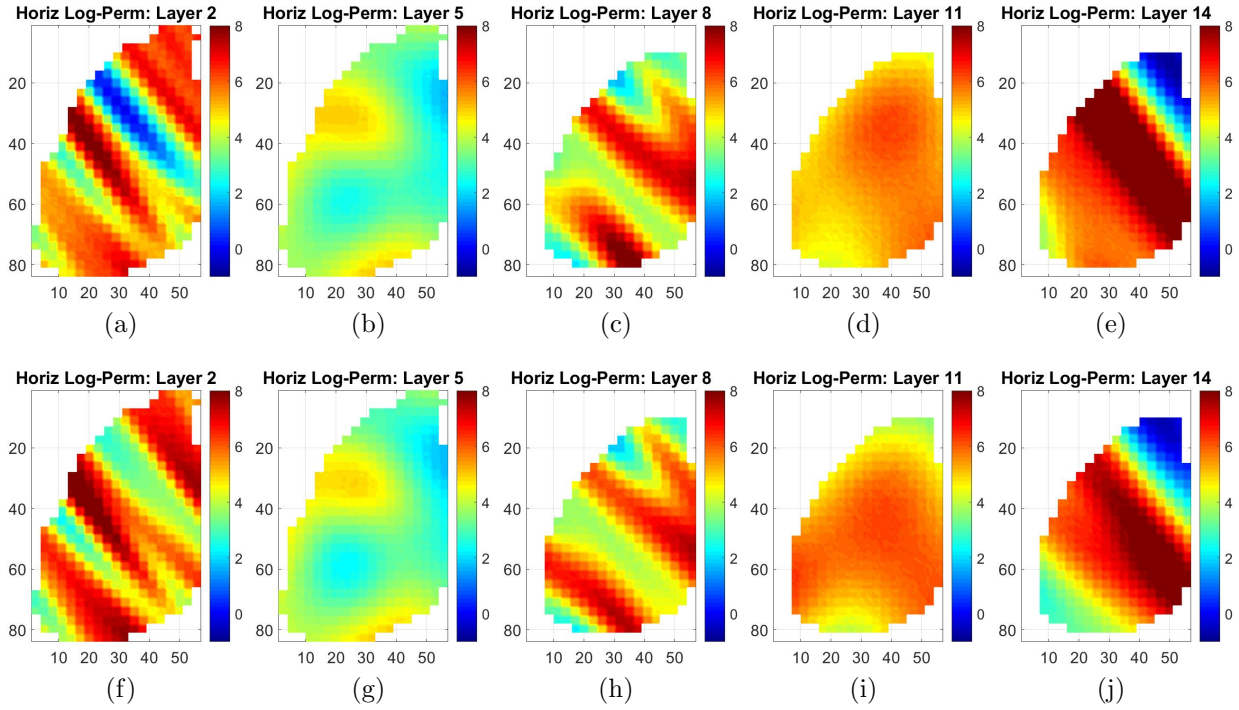


Figure 4.75: Horizontal log-permeability fields for the large scale case comparing the prior model (top row, layers 2, 5, 8, 11 and 14) with the history matched model (bottom row, layers 2, 5, 8, 11 and 14) for the converged model 2.

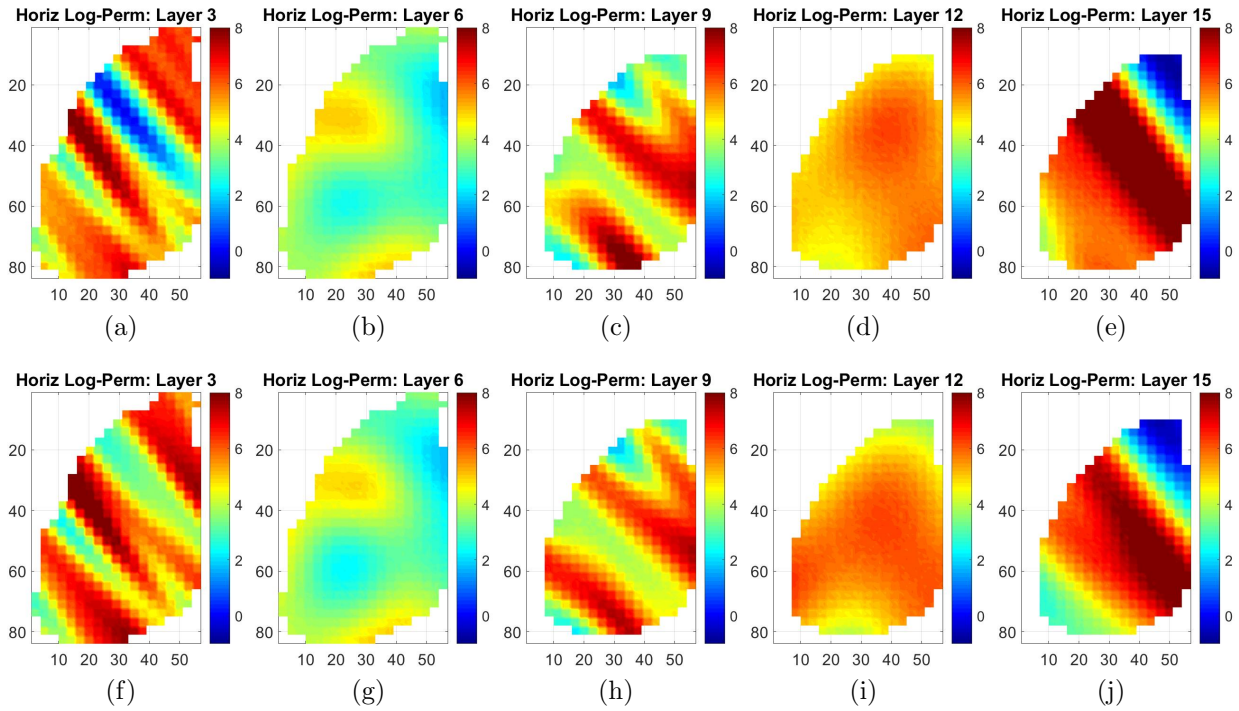


Figure 4.76: Horizontal log-permeability fields for the large scale case comparing the prior model (top row, layers 3, 6, 9, 12 and 15) with the history matched model (bottom row, layers 3, 6, 9, 12 and 15) for the converged model 2.

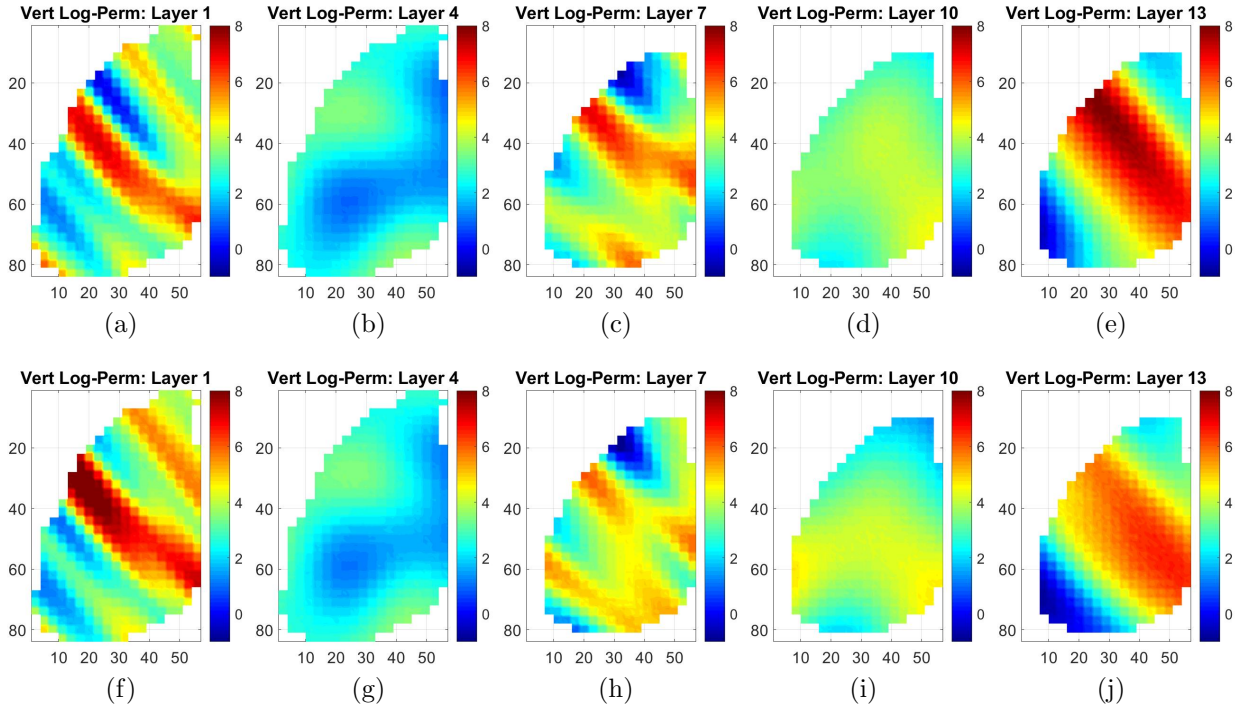


Figure 4.77: Vertical log-permeability fields for the large scale case comparing the prior model (top row, layers 1, 4, 7, 10 and 13) with the history matched model (bottom row, layers 1, 4, 7, 10 and 13) for the converged model 2.

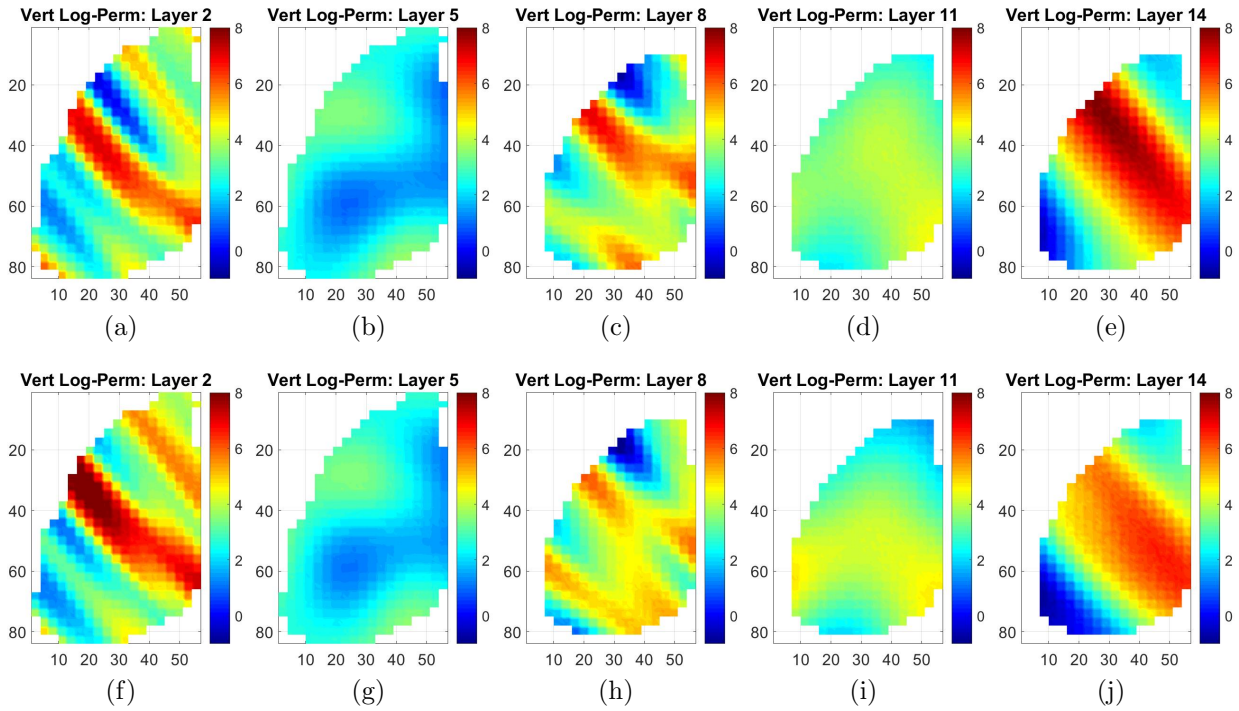


Figure 4.78: Vertical log-permeability fields for the large scale case comparing the prior model (top row, layers 2, 5, 8, 11 and 14) with the history matched model (bottom row, layers 2, 5, 8, 11 and 14) for the converged model 2.

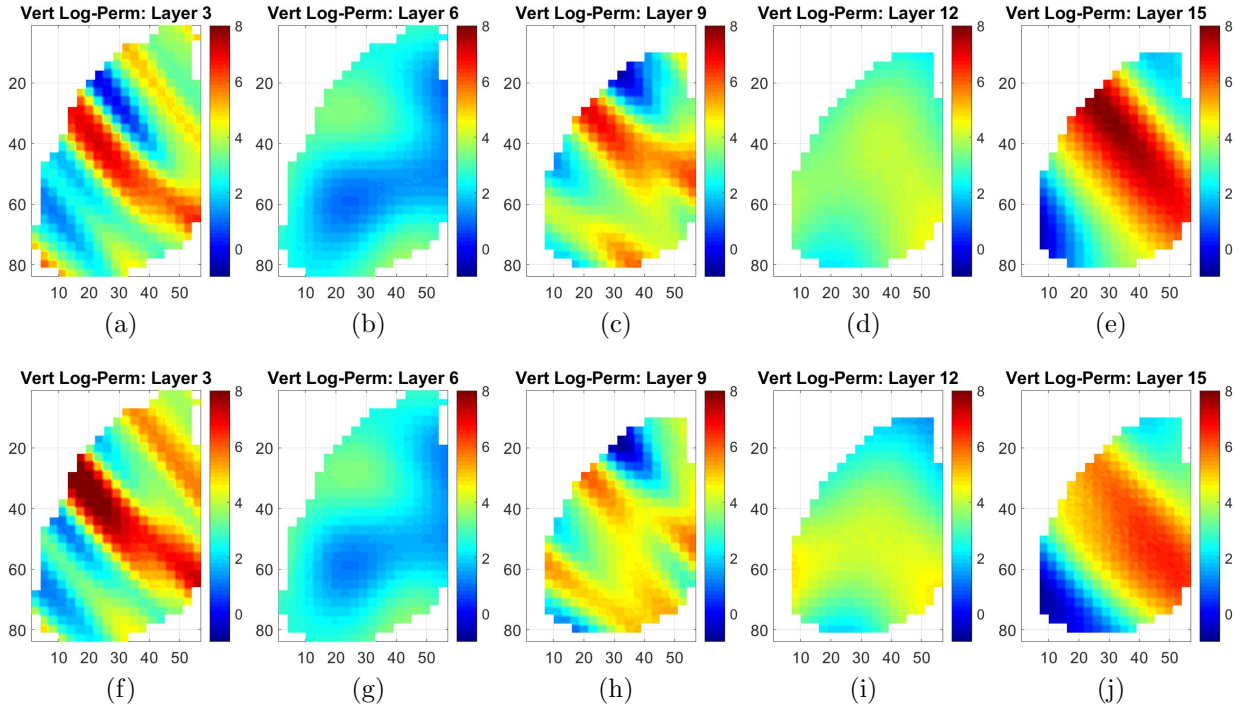


Figure 4.79: Vertical log-permeability fields for the large scale case comparing the prior model (top row, layers 3, 6, 9, 12 and 15) with the history matched model (bottom row, layers 3, 6, 9, 12 and 15) for the converged model 2.

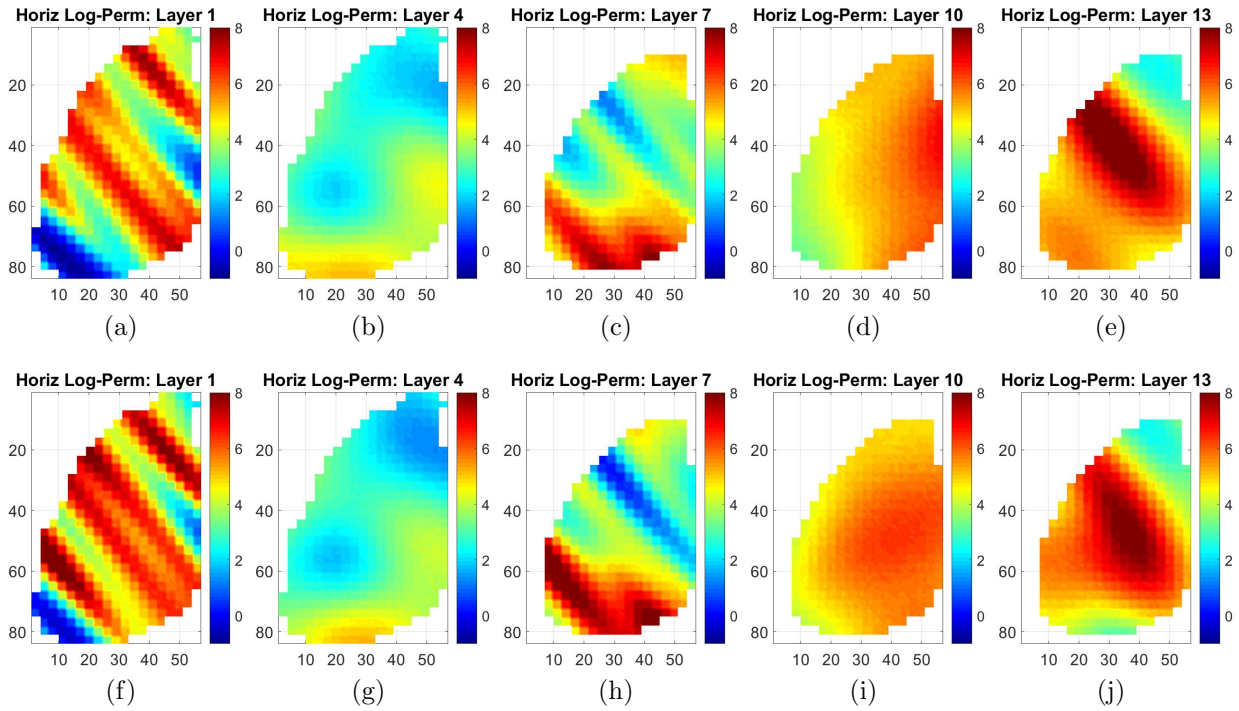


Figure 4.80: Horizontal log-permeability fields for the large scale case comparing the prior model (top row, layers 1, 4, 7, 10 and 13) with the history matched model (bottom row, layers 1, 4, 7, 10 and 13) for the converged model 3.

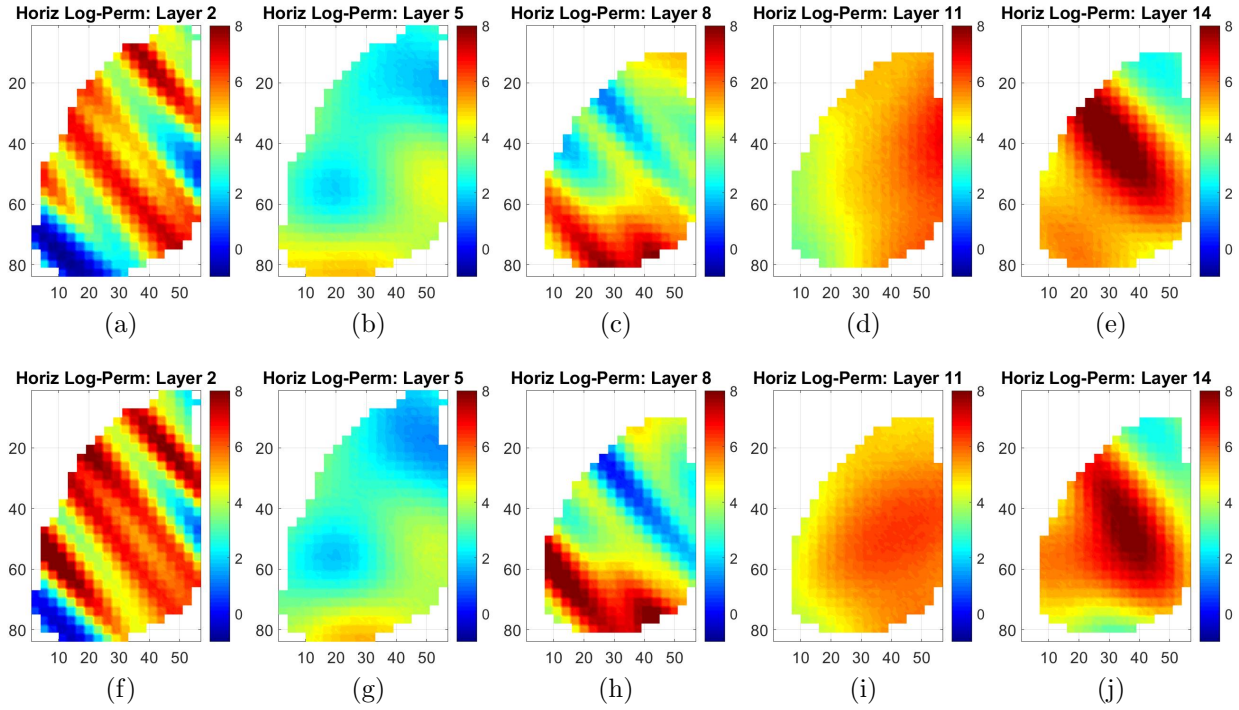


Figure 4.81: Horizontal log-permeability fields for the large scale case comparing the prior model (top row, layers 2, 5, 8, 11 and 14) with the history matched model (bottom row, layers 2, 5, 8, 11 and 14) for the converged model 3.

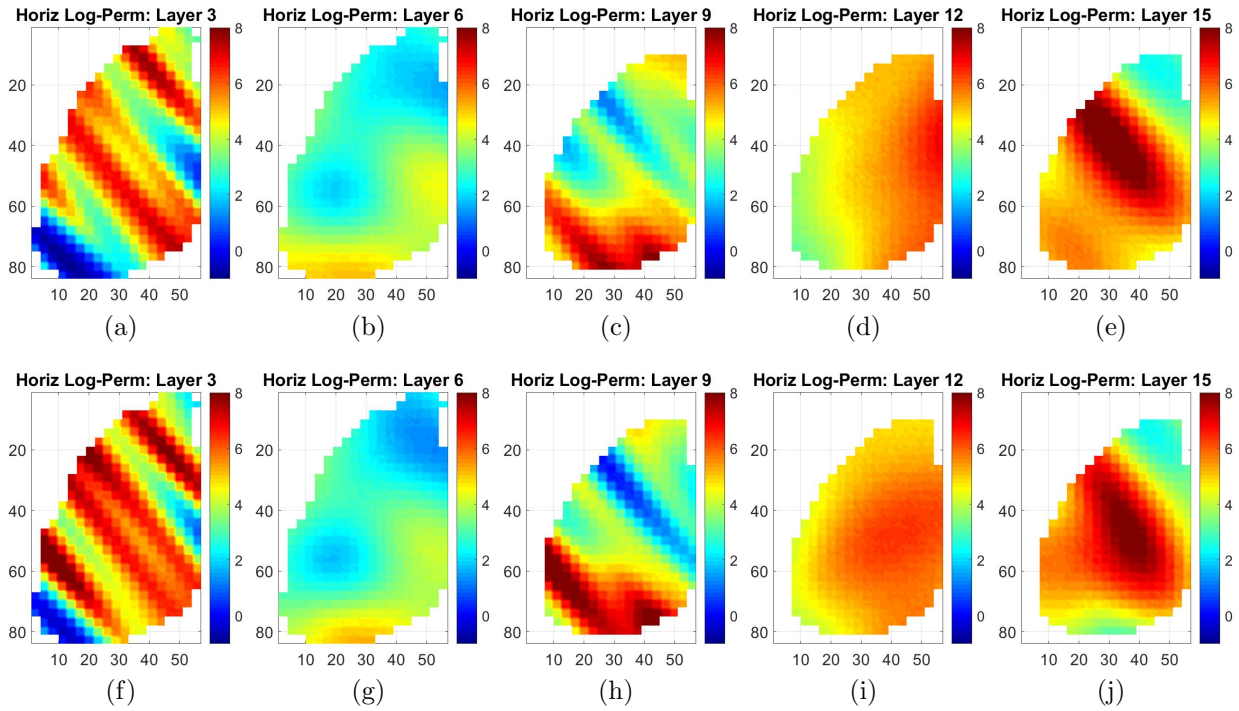


Figure 4.82: Horizontal log-permeability fields for the large scale case comparing the prior model (top row, layers 3, 6, 9, 12 and 15) with the history matched model (bottom row, layers 3, 6, 9, 12 and 15) for the converged model 3.

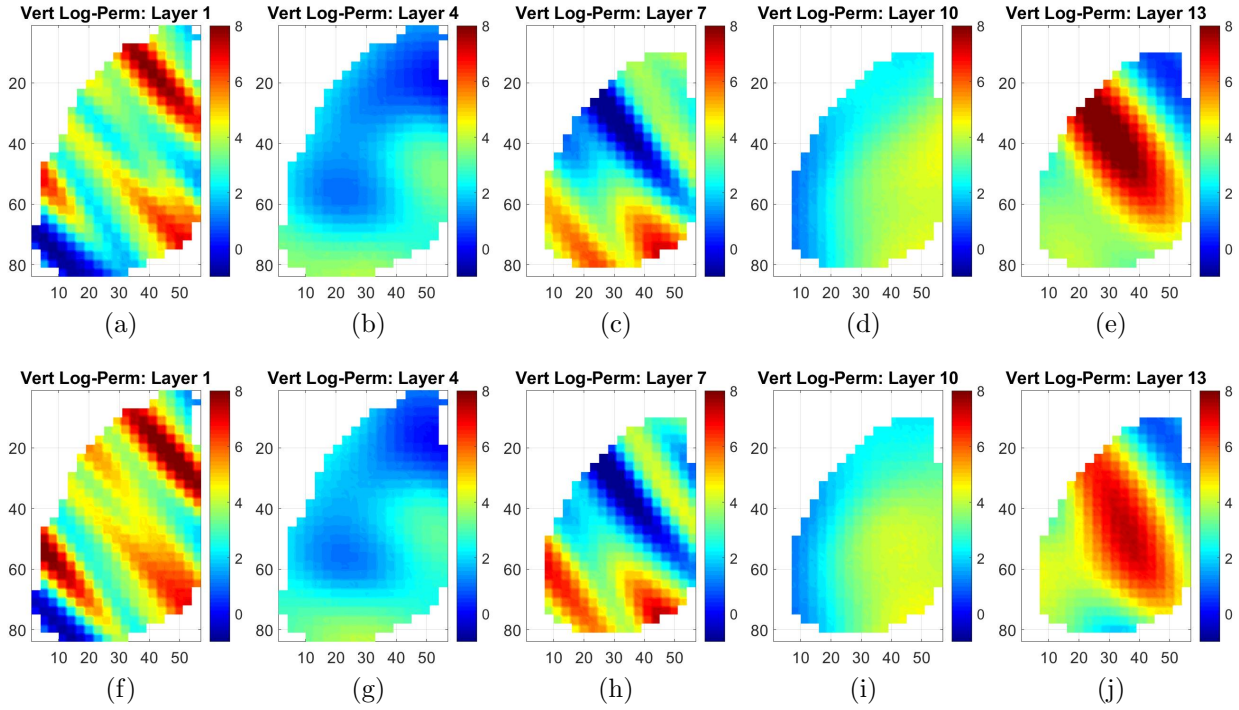


Figure 4.83: Vertical log-permeability fields for the large scale case comparing the prior model (top row, layers 1, 4, 7, 10 and 13) with the history matched model (bottom row, layers 1, 4, 7, 10 and 13) for the converged model 3.

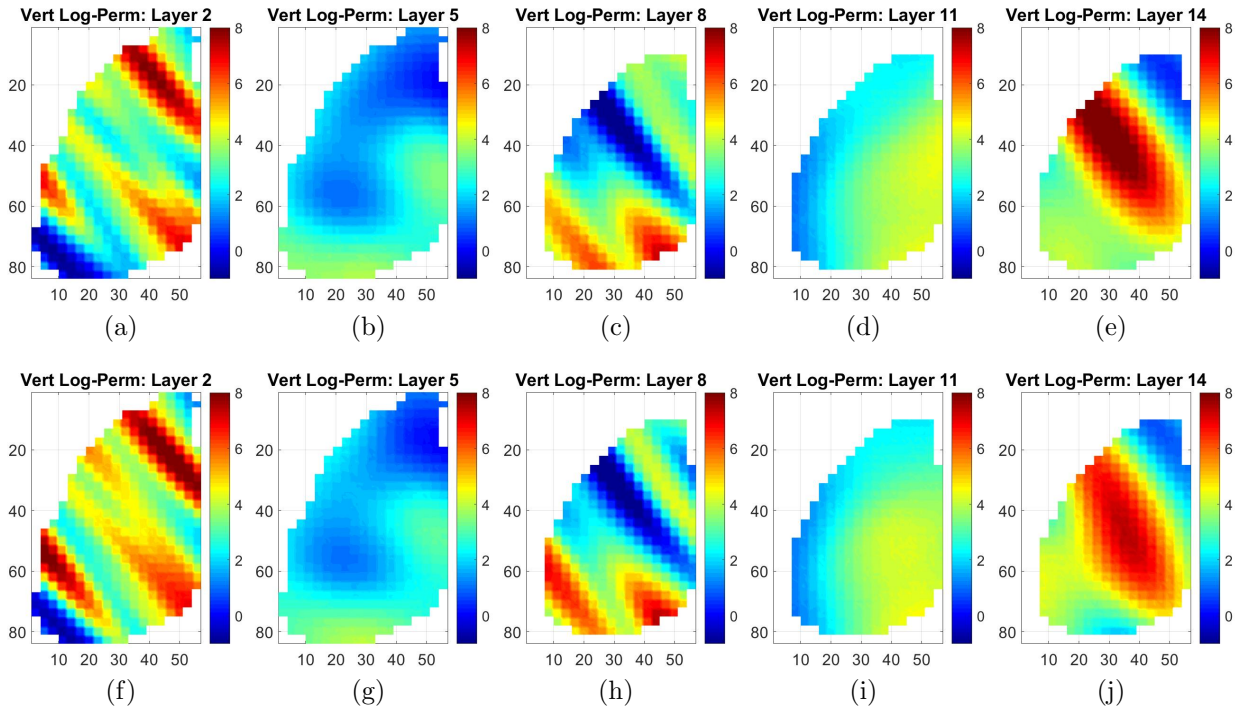


Figure 4.84: Vertical log-permeability fields for the large scale case comparing the prior model (top row, layers 2, 5, 8, 11 and 14) with the history matched model (bottom row, layers 2, 5, 8, 11 and 14) for the converged model 3.

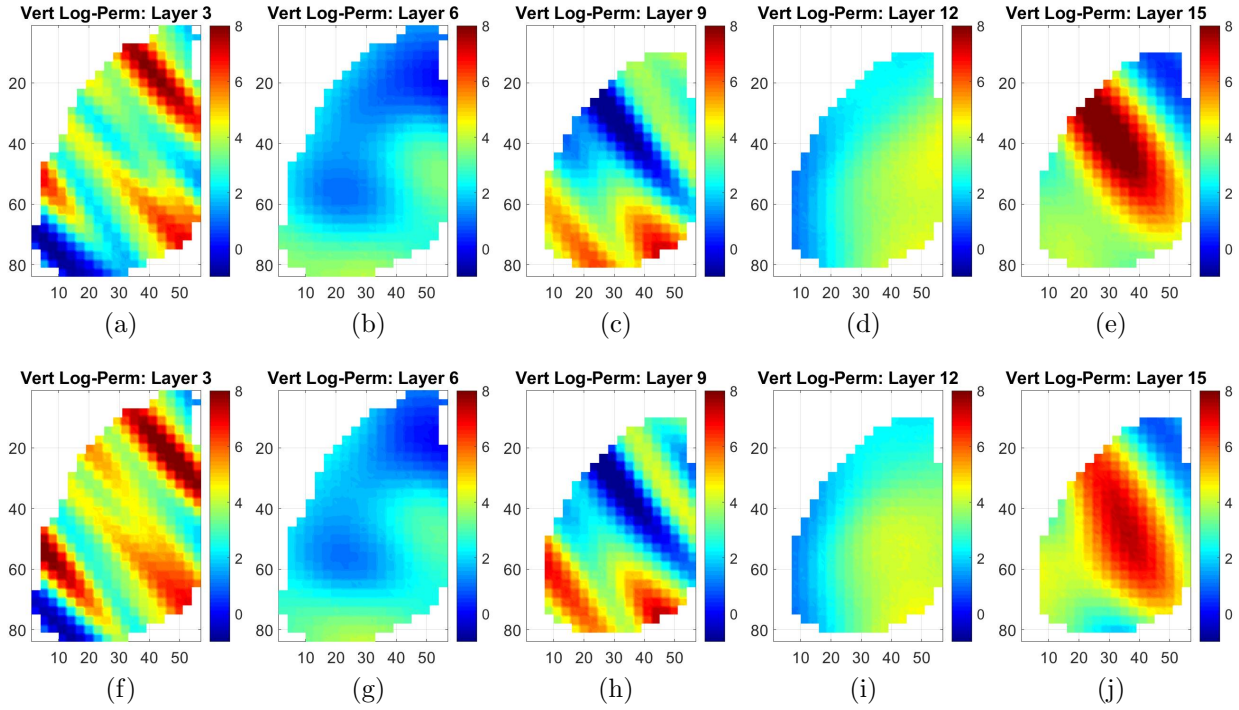


Figure 4.85: Vertical log-permeability fields for the large scale case comparing the prior model (top row, layers 3, 6, 9, 12 and 15) with the history matched model (bottom row, layers 3, 6, 9, 12 and 15) for the converged model 3.

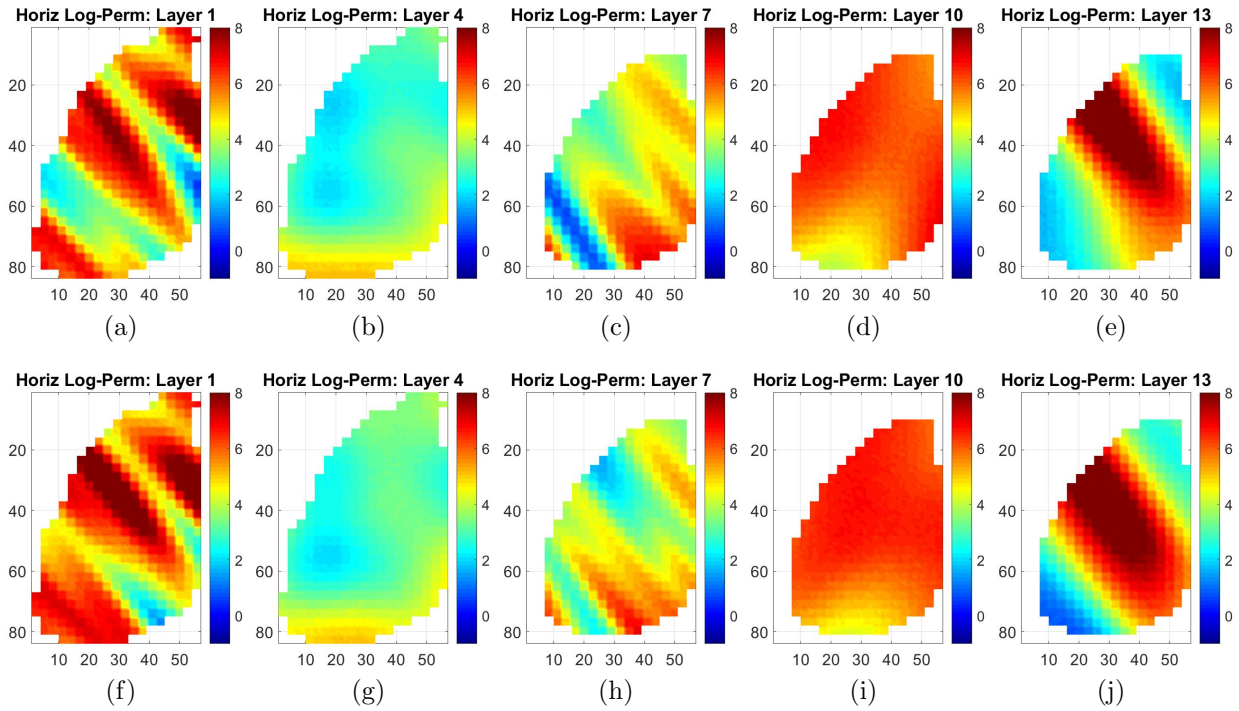


Figure 4.86: Horizontal log-permeability fields for the large scale case comparing the prior model (top row, layers 1, 4, 7, 10 and 13) with the history matched model (bottom row, layers 1, 4, 7, 10 and 13) for the converged model 4.

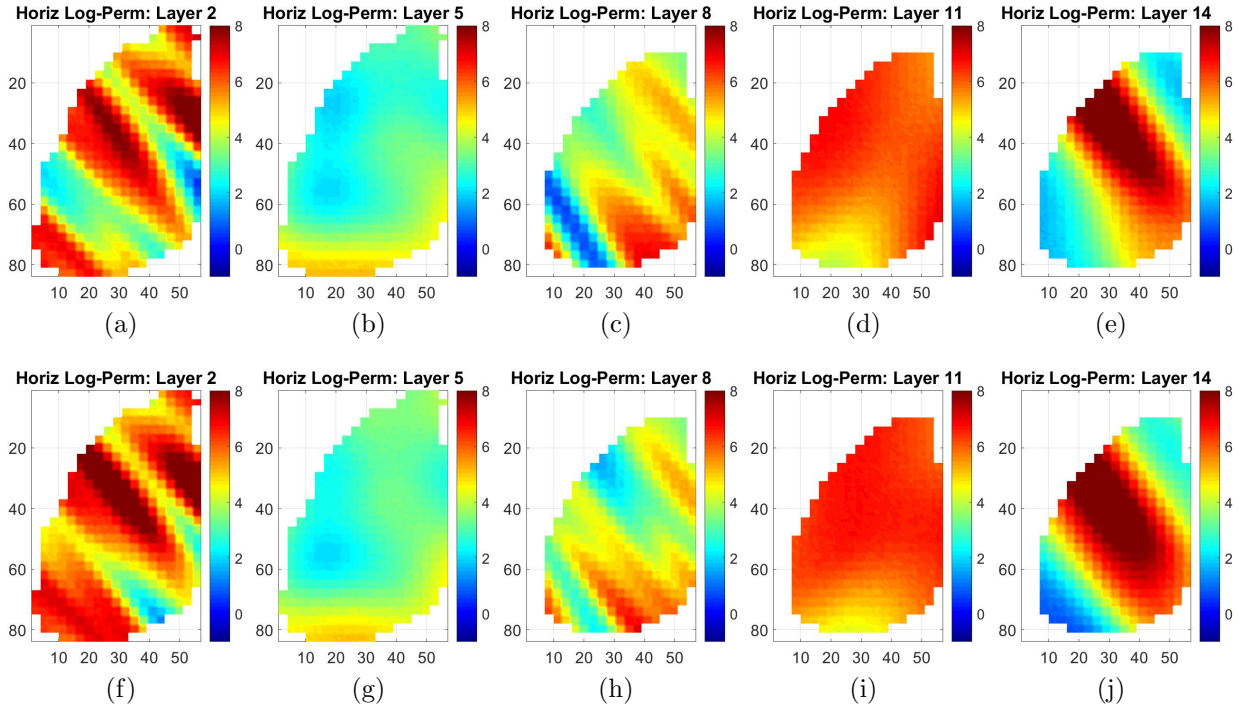


Figure 4.87: Horizontal log-permeability fields for the large scale case comparing the prior model (top row, layers 2, 5, 8, 11 and 14) with the history matched model (bottom row, layers 2, 5, 8, 11 and 14) for the converged model 4.

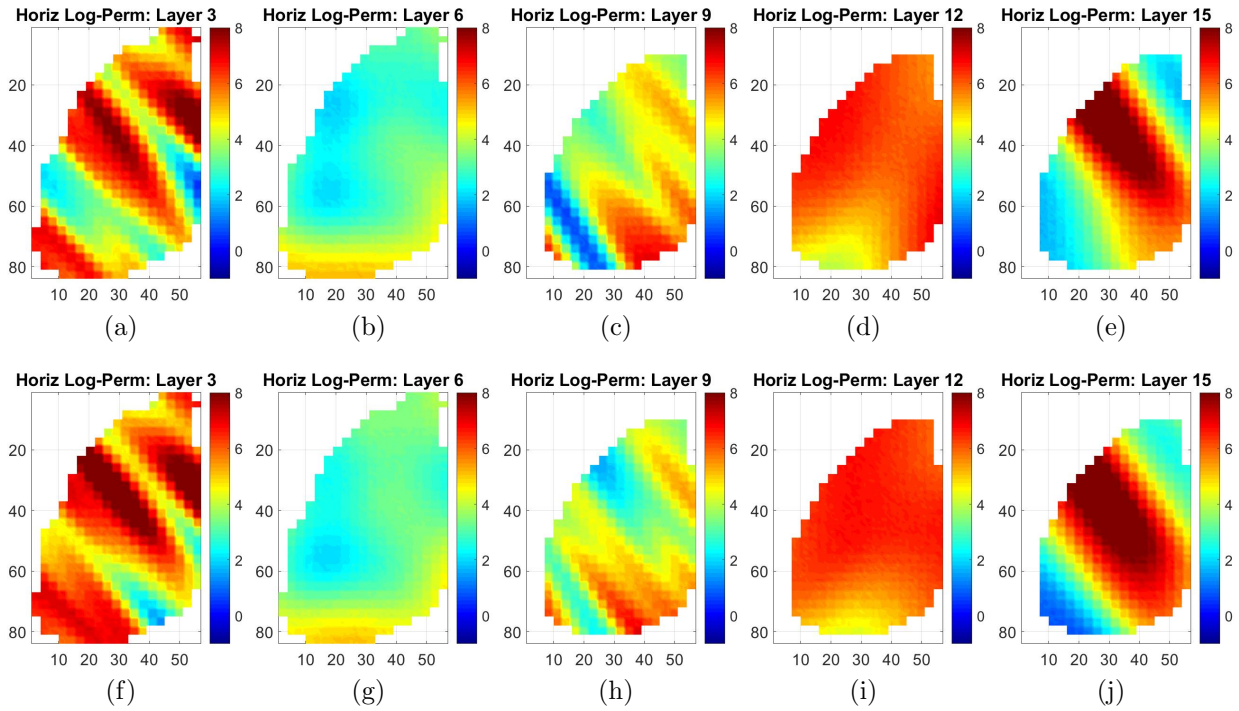


Figure 4.88: Horizontal log-permeability fields for the large scale case comparing the prior model (top row, layers 3, 6, 9, 12 and 15) with the history matched model (bottom row, layers 3, 6, 9, 12 and 15) for the converged model 4.

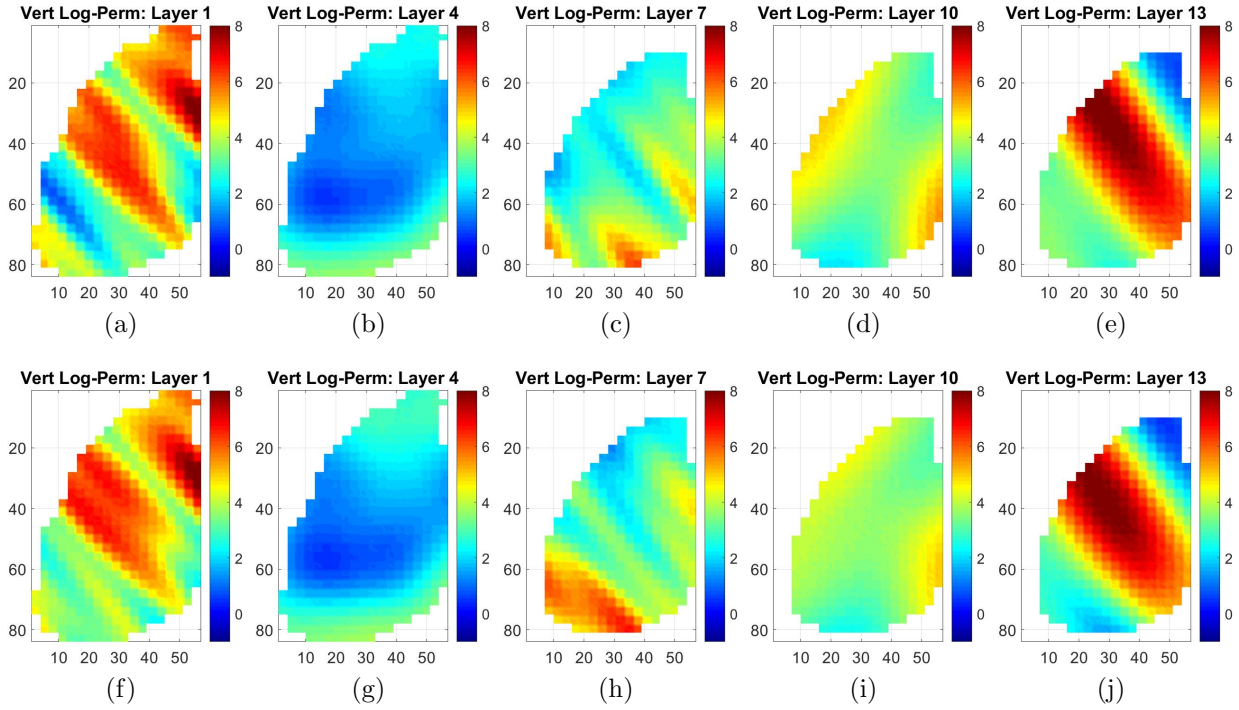


Figure 4.89: Vertical log-permeability fields for the large scale case comparing the prior model (top row, layers 1, 4, 7, 10 and 13) with the history matched model (bottom row, layers 1, 4, 7, 10 and 13) for the converged model 4.

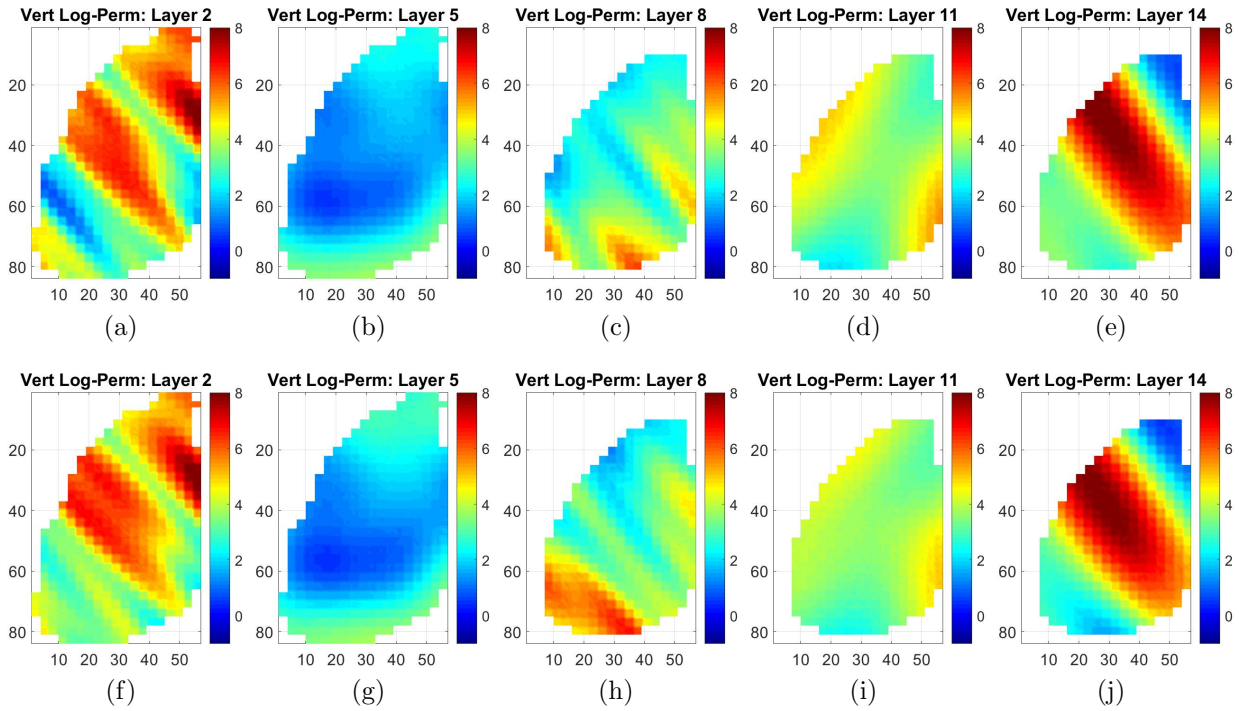


Figure 4.90: Vertical log-permeability fields for the large scale case comparing the prior model (top row, layers 2, 5, 8, 11 and 14) with the history matched model (bottom row, layers 2, 5, 8, 11 and 14) for the converged model 4.

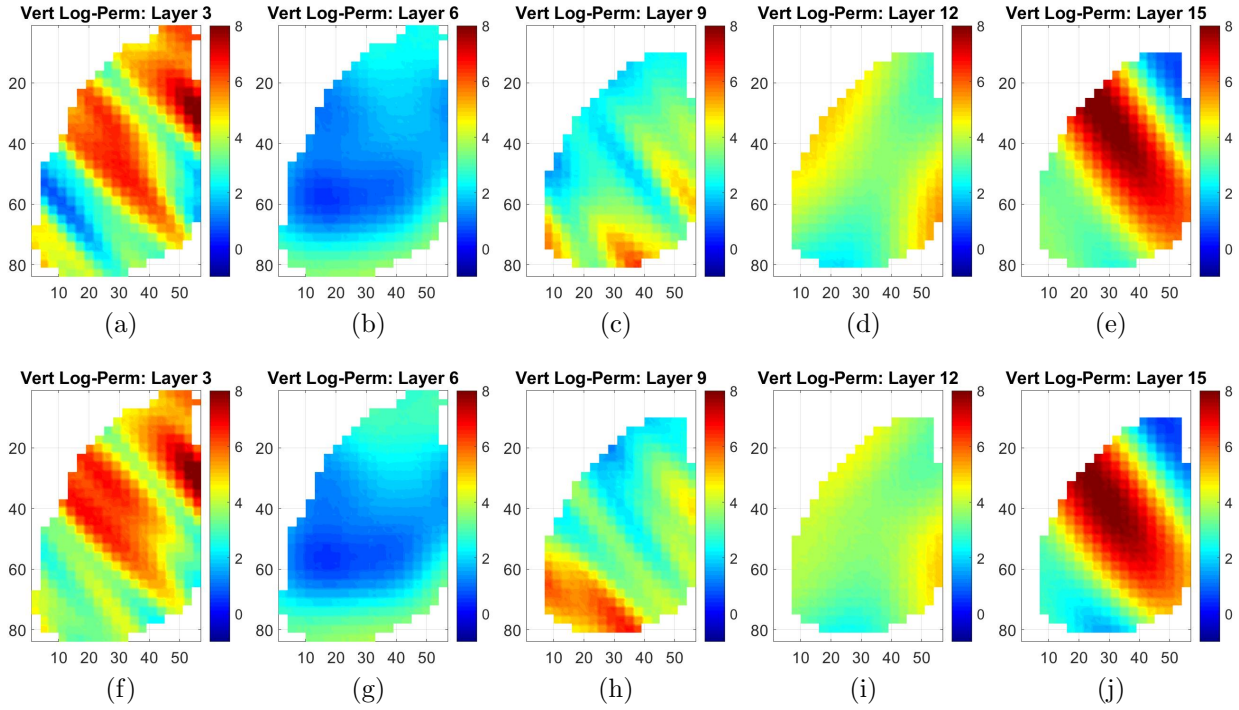


Figure 4.91: Vertical log-permeability fields for the large scale case comparing the prior model (top row, layers 3, 6, 9, 12 and 15) with the history matched model (bottom row, layers 3, 6, 9, 12 and 15) for the converged model 4.

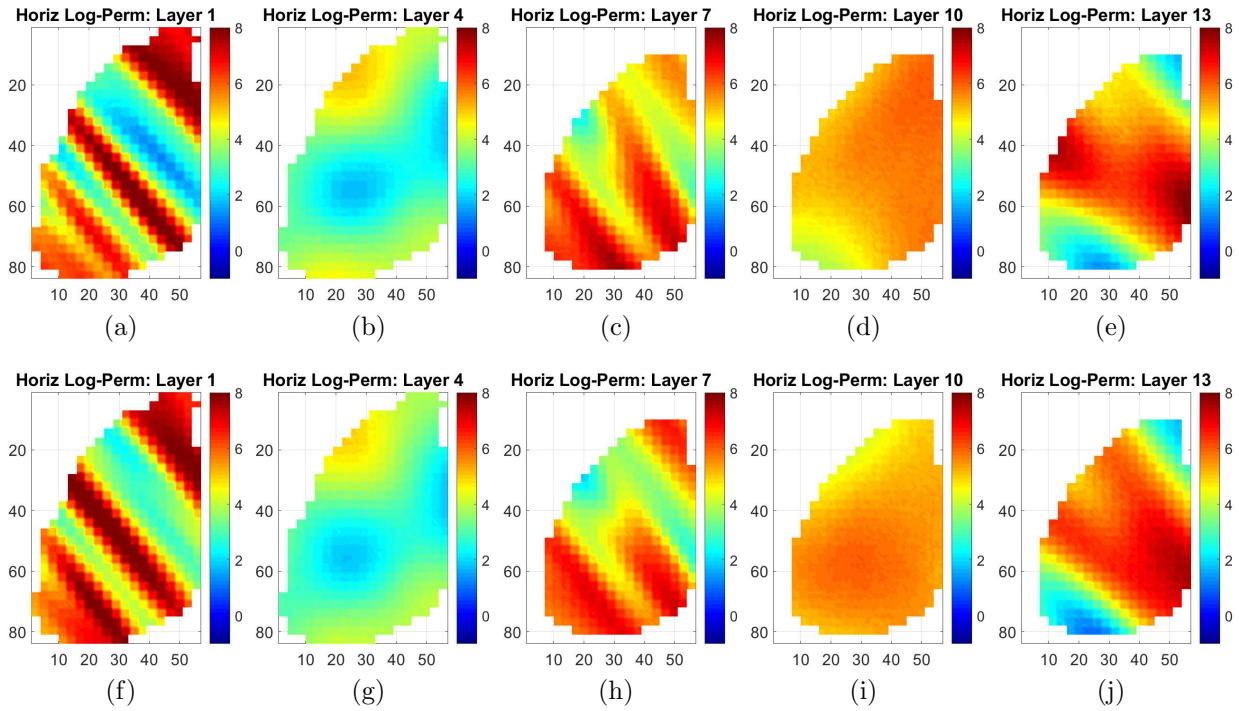


Figure 4.92: Horizontal log-permeability fields for the large scale case comparing the prior model (top row, layers 1, 4, 7, 10 and 13) with the history matched model (bottom row, layers 1, 4, 7, 10 and 13) for the converged model 5.

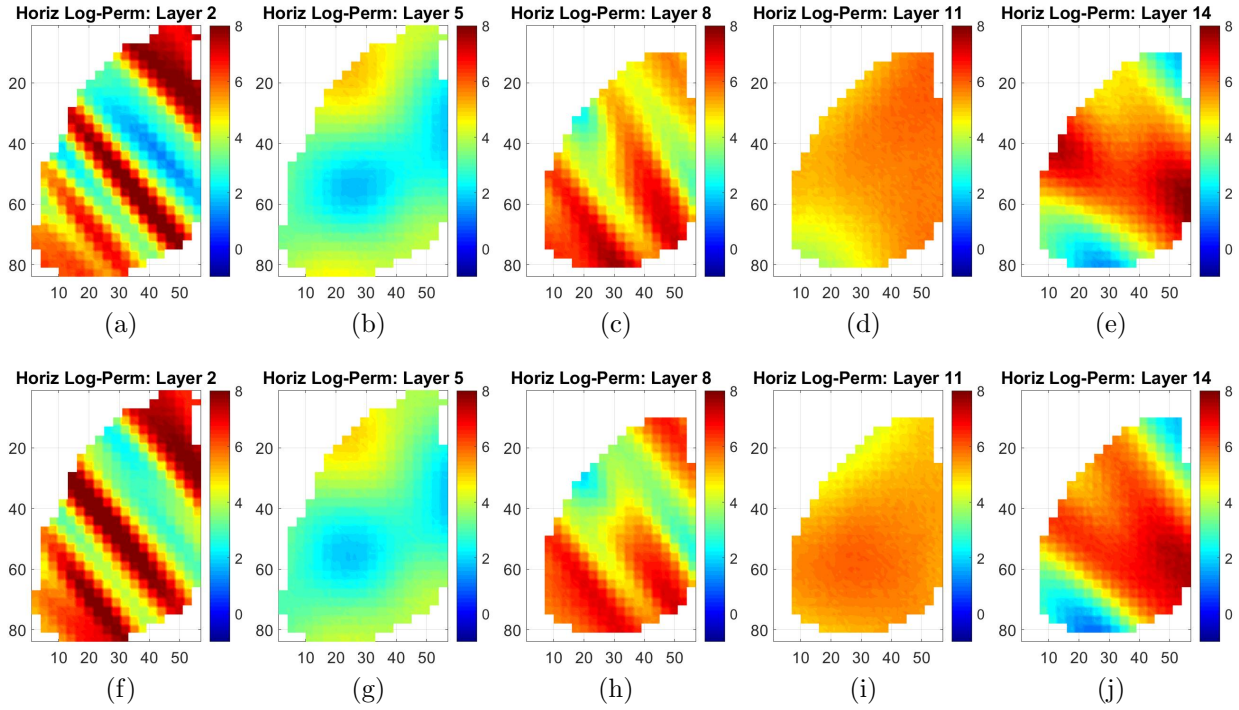


Figure 4.93: Horizontal log-permeability fields for the large scale case comparing the prior model (top row, layers 2, 5, 8, 11 and 14) with the history matched model (bottom row, layers 2, 5, 8, 11 and 14) for the converged model 5.

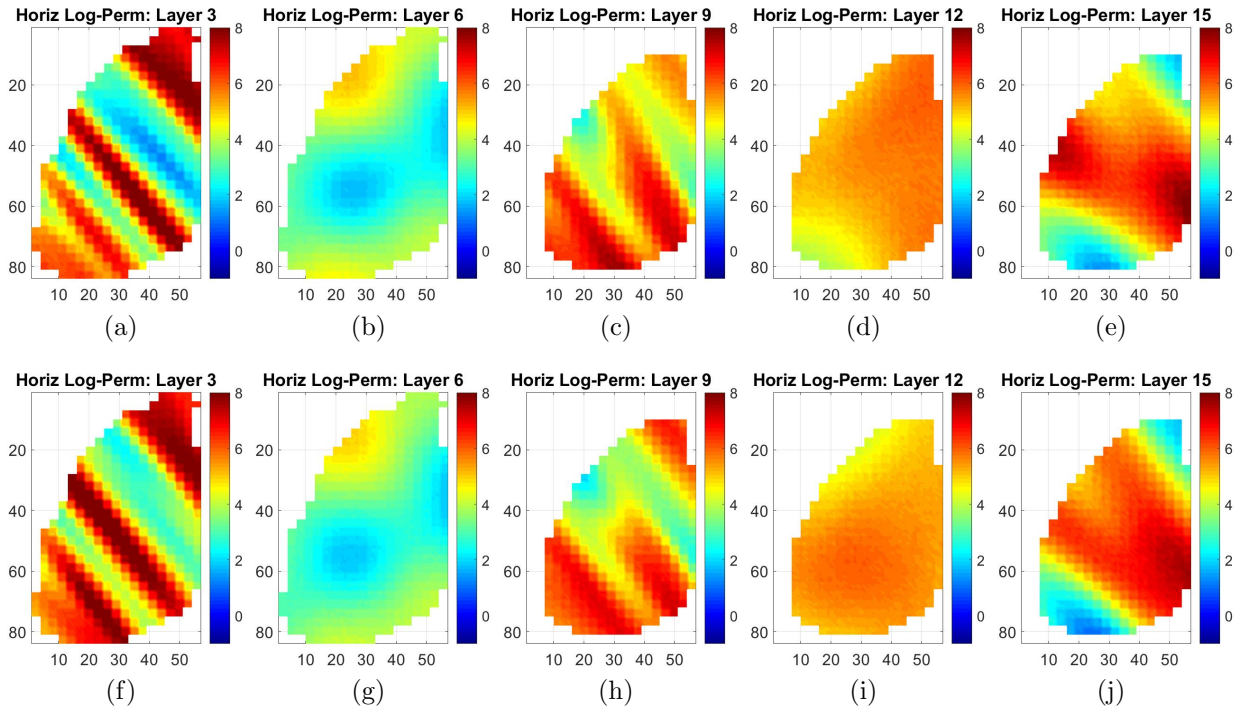


Figure 4.94: Horizontal log-permeability fields for the large scale case comparing the prior model (top row, layers 3, 6, 9, 12 and 15) with the history matched model (bottom row, layers 3, 6, 9, 12 and 15) for the converged model 5.

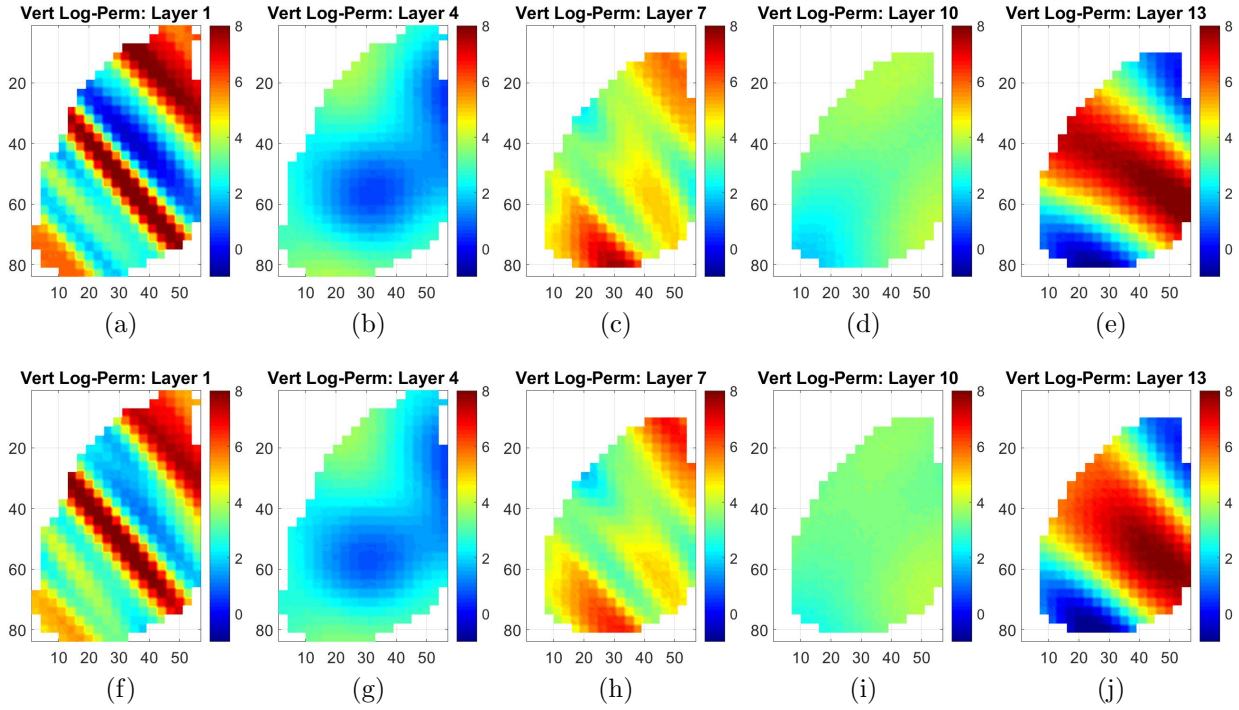


Figure 4.95: Vertical log-permeability fields for the large scale case comparing the prior model (top row, layers 1, 4, 7, 10 and 13) with the history matched model (bottom row, layers 1, 4, 7, 10 and 13) for the converged model 5.

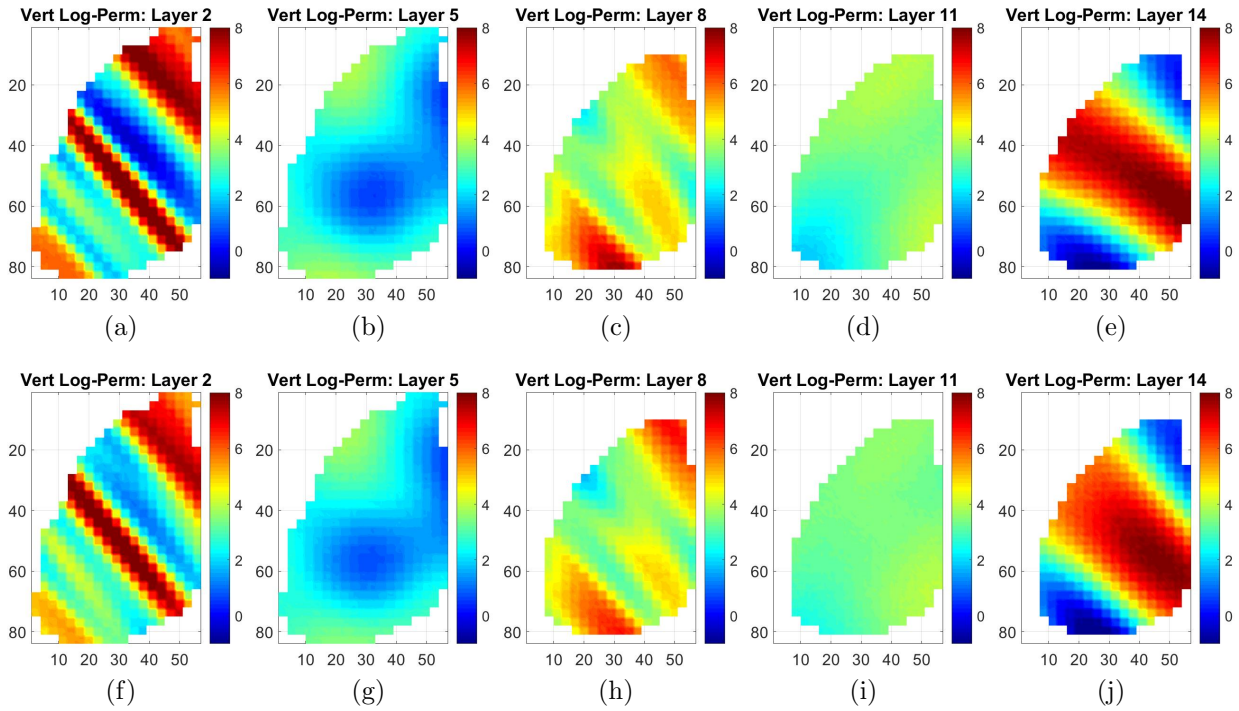


Figure 4.96: Vertical log-permeability fields for the large scale case comparing the prior model (top row, layers 2, 5, 8, 11 and 14) with the history matched model (bottom row, layers 2, 5, 8, 11 and 14) for the converged model 5.

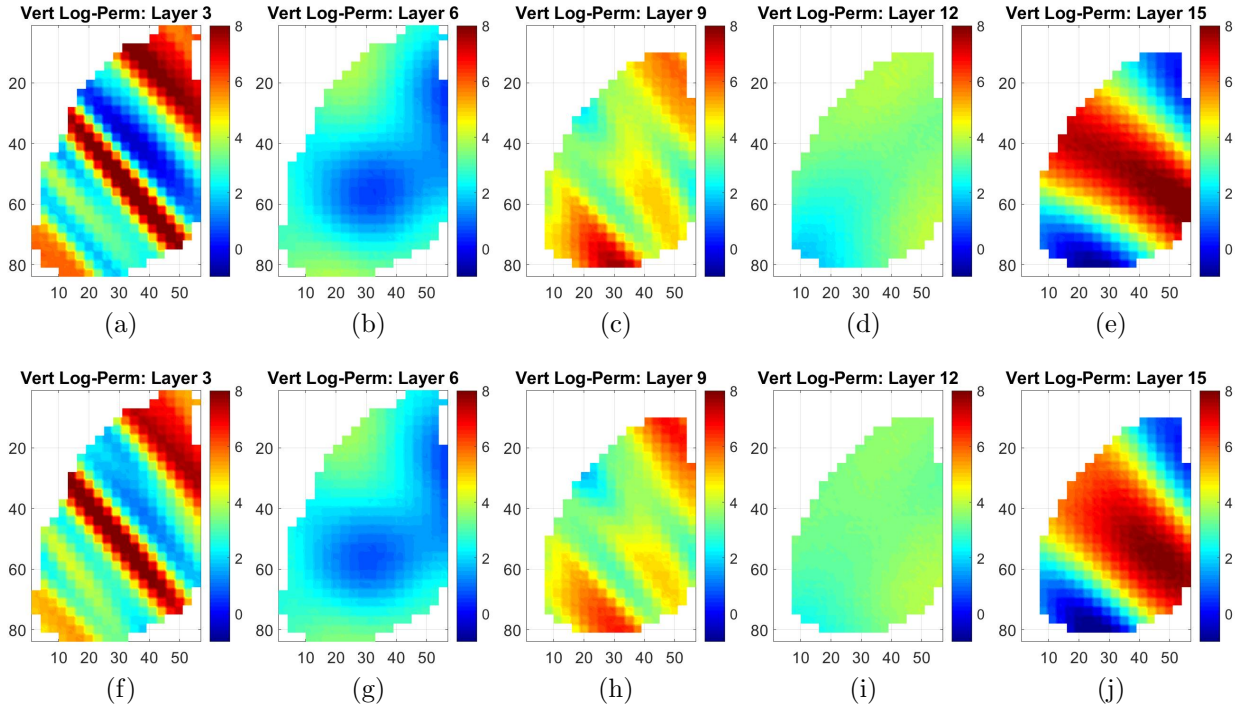


Figure 4.97: Vertical log-permeability fields for the large scale case comparing the prior model (top row, layers 3, 6, 9, 12 and 15) with the history matched model (bottom row, layers 3, 6, 9, 12 and 15) for the converged model 5.

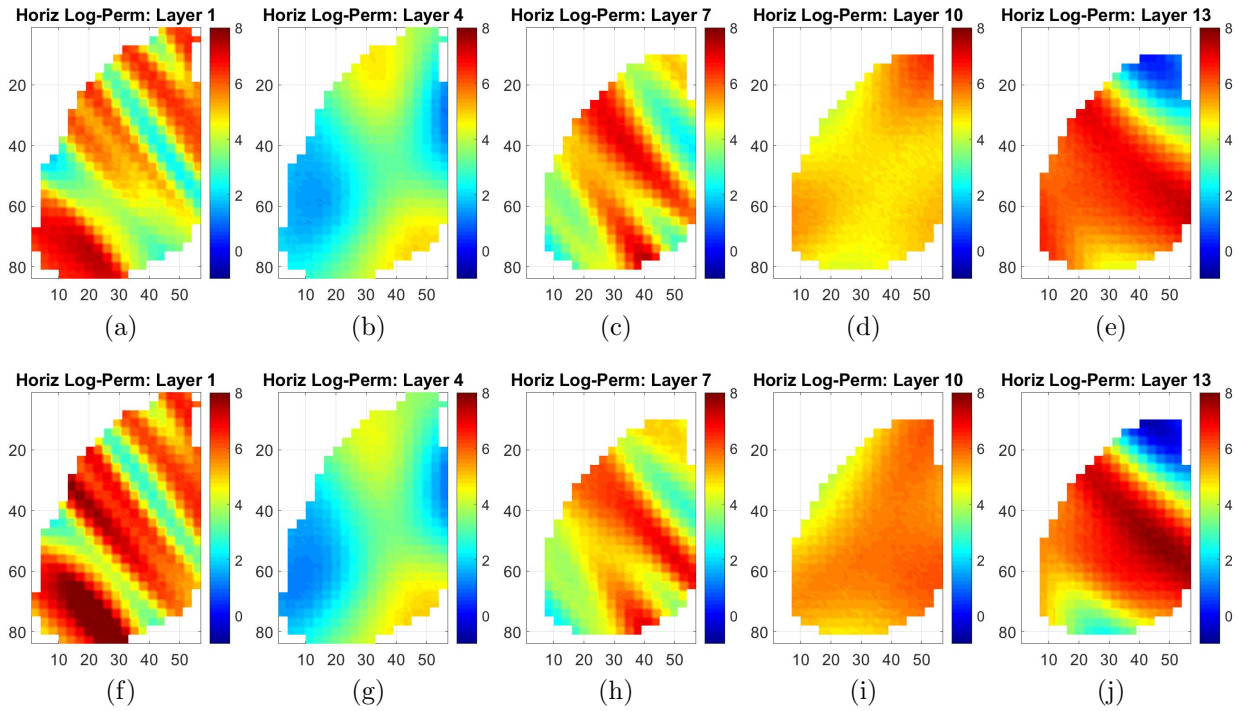


Figure 4.98: Horizontal log-permeability fields for the large scale case comparing the prior model (top row, layers 1, 4, 7, 10 and 13) with the history matched model (bottom row, layers 1, 4, 7, 10 and 13) for the converged model 6.

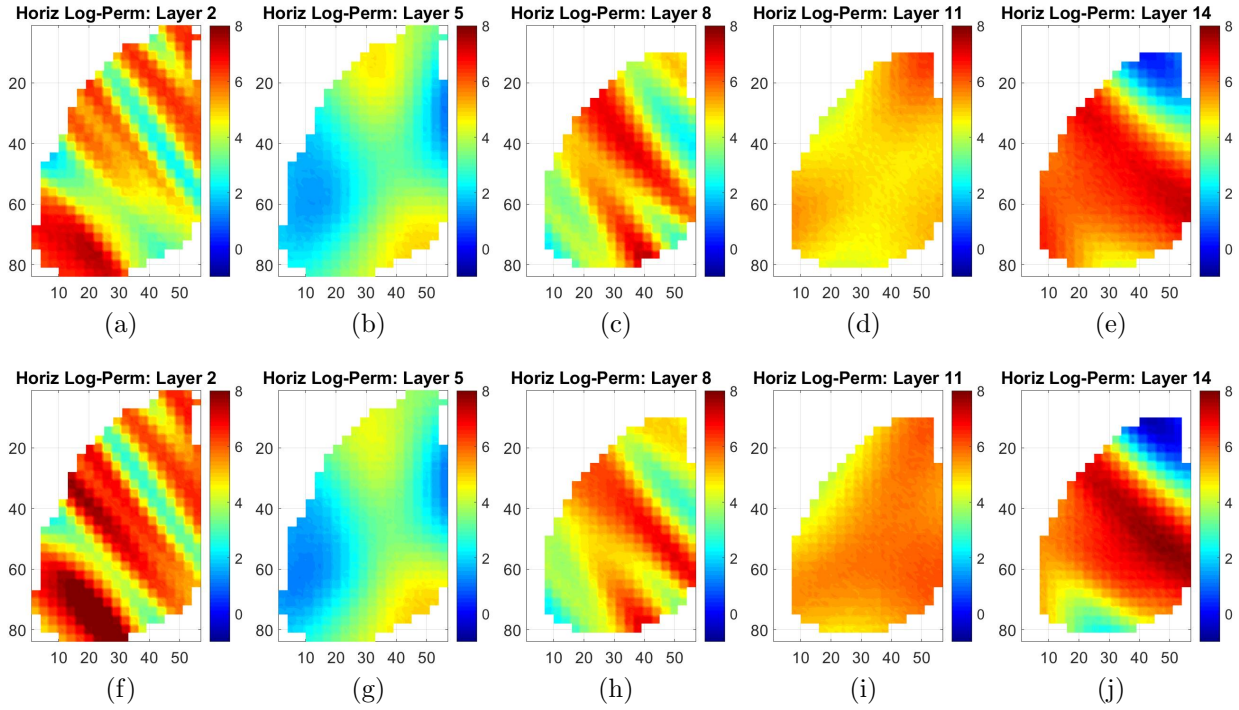


Figure 4.99: Horizontal log-permeability fields for the large scale case comparing the prior model (top row, layers 2, 5, 8, 11 and 14) with the history matched model (bottom row, layers 2, 5, 8, 11 and 14) for the converged model 6.

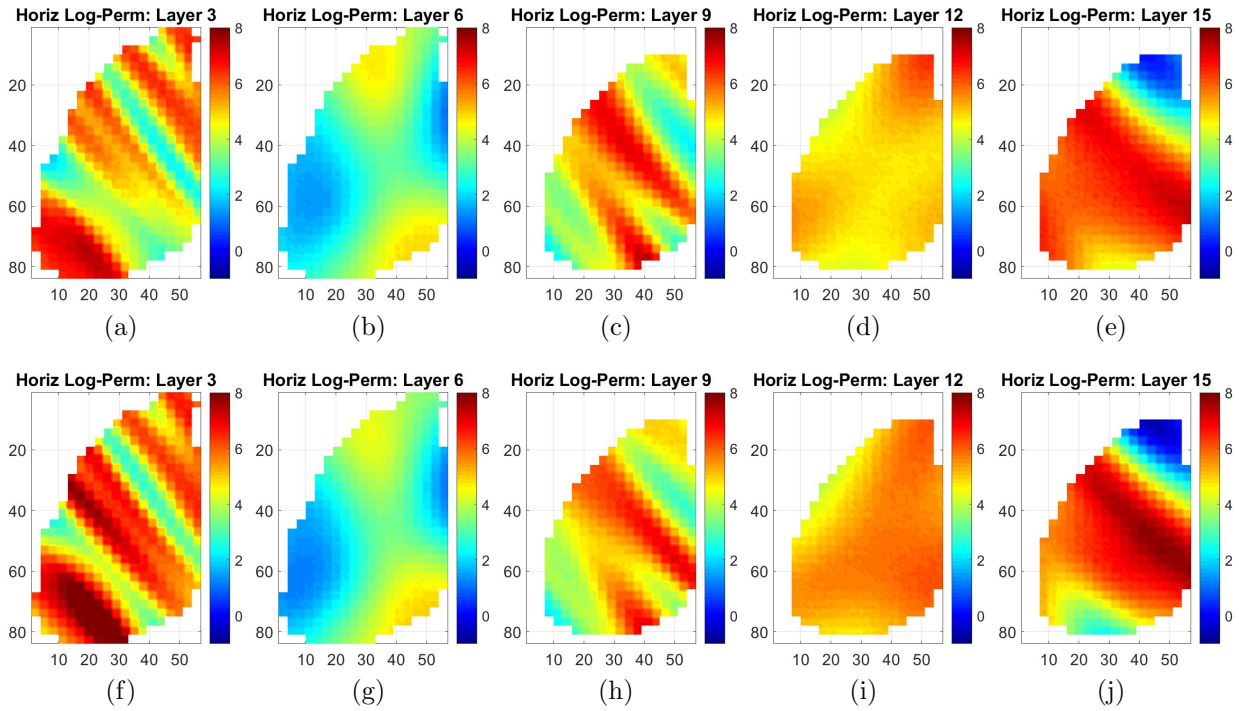


Figure 4.100: Horizontal log-permeability fields for the large scale case comparing the prior model (top row, layers 3, 6, 9, 12 and 15) with the history matched model (bottom row, layers 3, 6, 9, 12 and 15) for the converged model 6.

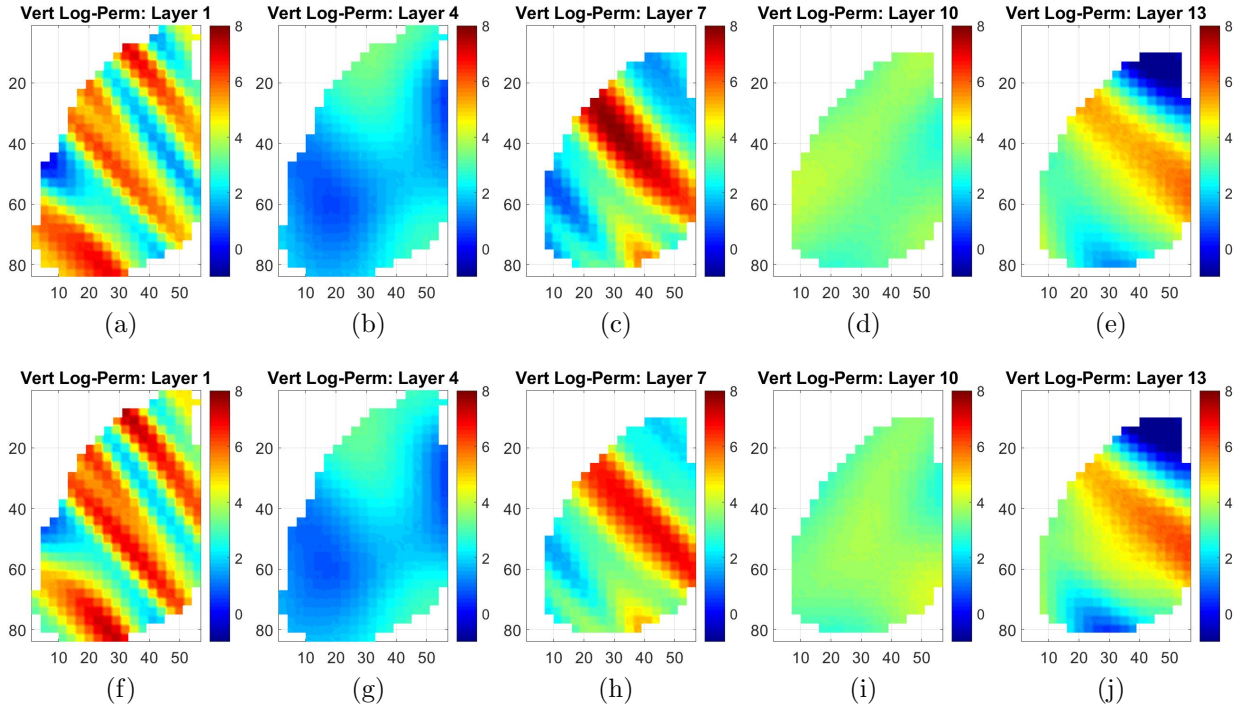


Figure 4.101: Vertical log-permeability fields for the large scale case comparing the prior model (top row, layers 1, 4, 7, 10 and 13) with the history matched model (bottom row, layers 1, 4, 7, 10 and 13) for the converged model 6.

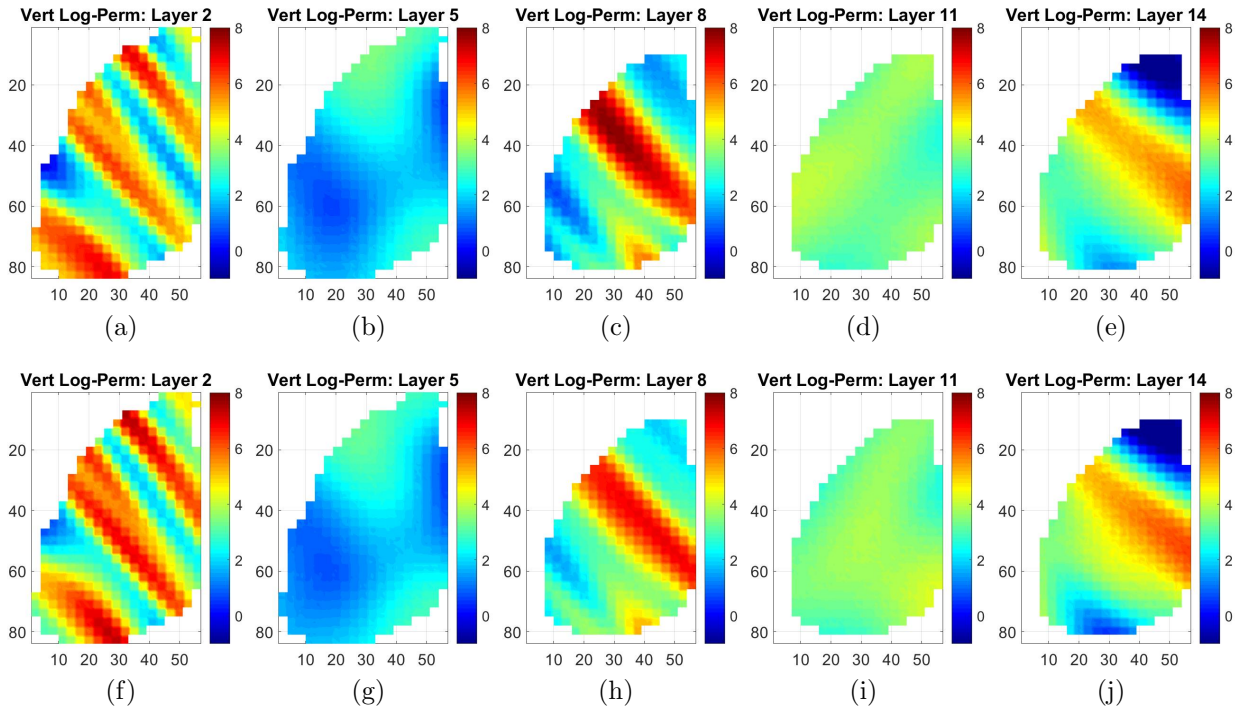


Figure 4.102: Vertical log-permeability fields for the large scale case comparing the prior model (top row, layers 2, 5, 8, 11 and 14) with the history matched model (bottom row, layers 2, 5, 8, 11 and 14) for the converged model 6.

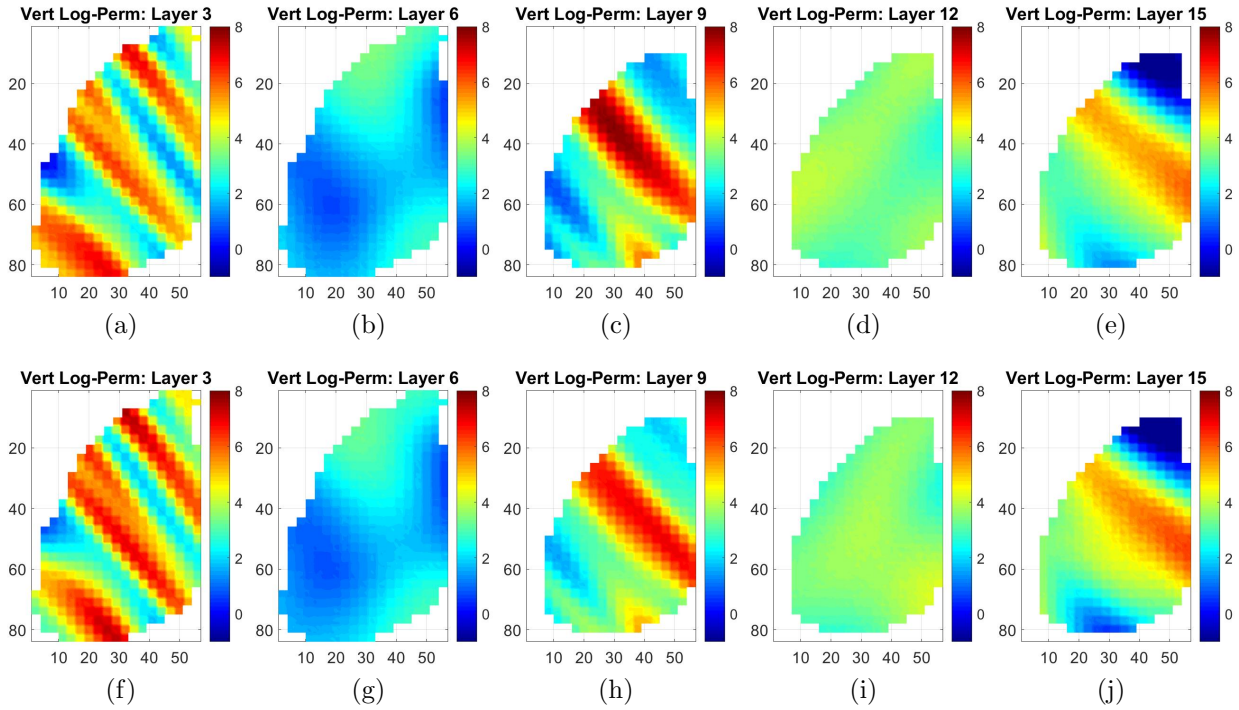


Figure 4.103: Vertical log-permeability fields for the large scale case comparing the prior model (top row, layers 3, 6, 9, 12 and 15) with the history matched model (bottom row, layers 3, 6, 9, 12 and 15) for the converged model 6.

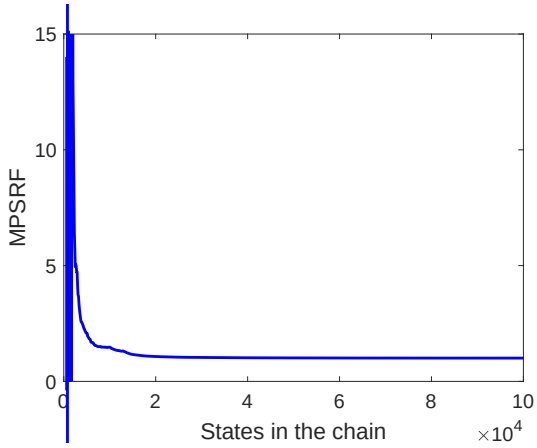


Figure 4.104: MCMC sampling convergence based on MPSRF for the large scale case.

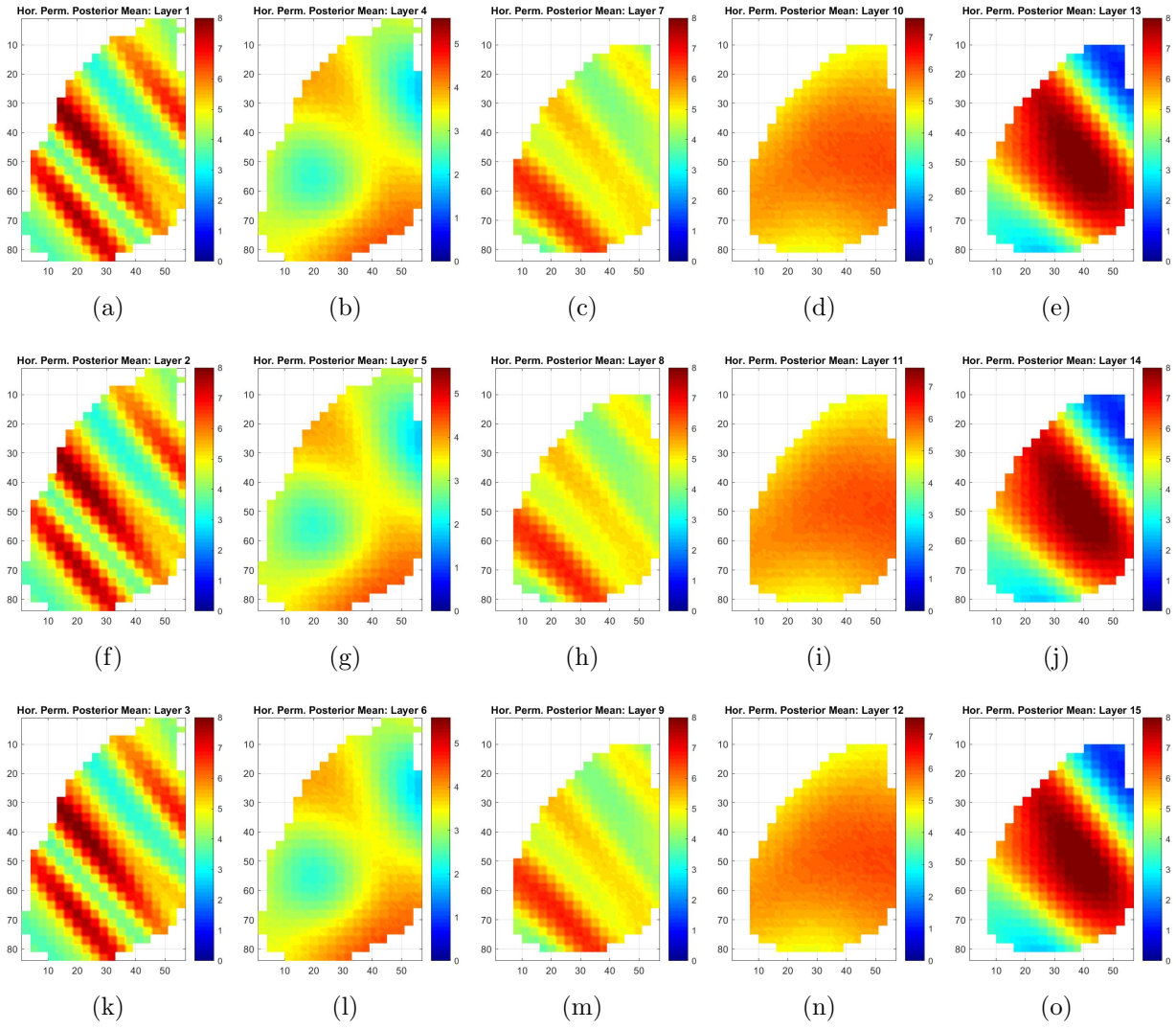


Figure 4.105: Posterior mean for horizontal log-permeability for the large scale case.

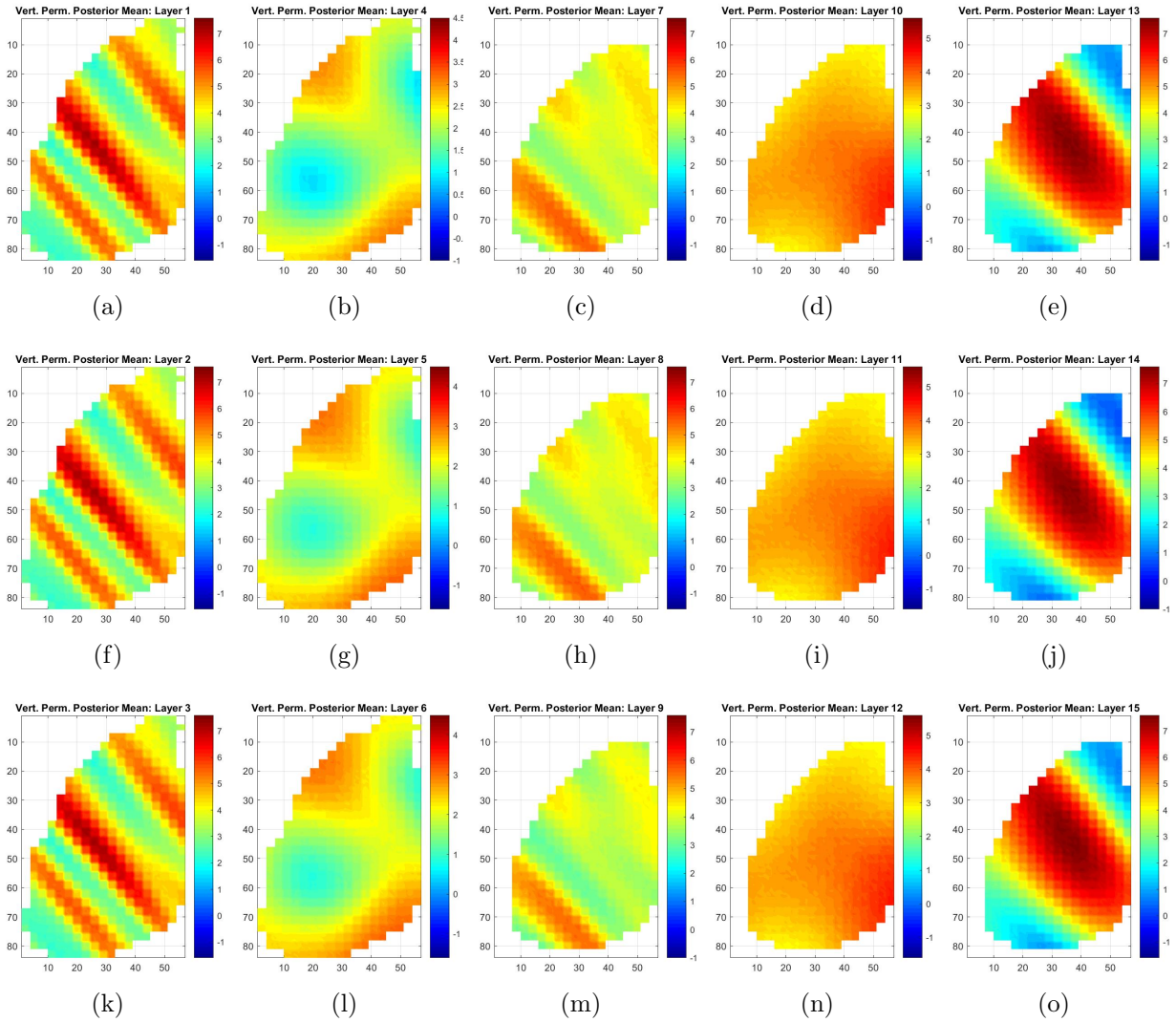


Figure 4.106: Posterior mean for vertical log-permeability for the large scale case.

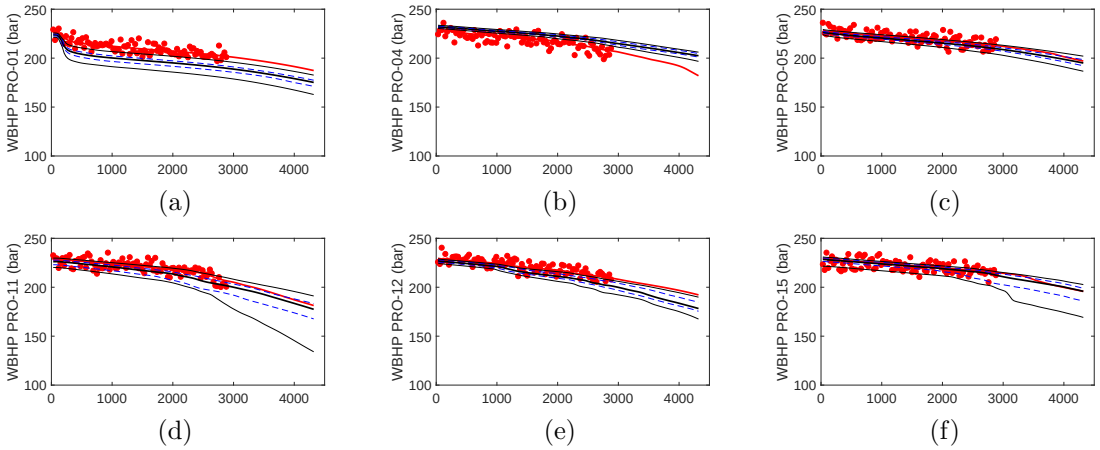


Figure 4.107: Marginal distribution for well bottom hole pressure for producers PRO-01 through PRO-15. In all figures, we show the corresponding true value (solid thick red curve), the mean of the marginal distribution (solid thick black curve), the percentiles P25 and P75 of the marginal distribution (two blue dashed curves), and the percentiles P5 and P95 of the marginal distribution (two thin black curves). To generate the marginal distributions presented here, all states after state 40,000 in the five chains are used, which gives a total of 300,000 states.

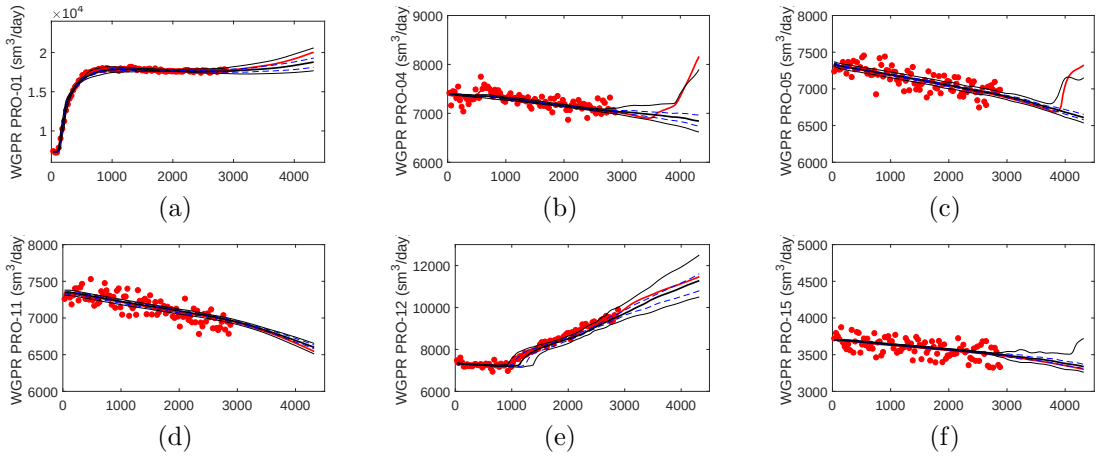


Figure 4.108: Marginal distribution for well gas production rate for producers PRO-01 through PRO-15. In all figures, we show the corresponding true value (solid thick red curve), the mean of the marginal distribution (solid thick black curve), the percentiles P25 and P75 of the marginal distribution (two blue dashed curves), and the percentiles P5 and P95 of the marginal distribution (two thin black curves). To generate the marginal distributions presented here, all states after state 40,000 in the five chains are used, which gives a total of 300,000 states.

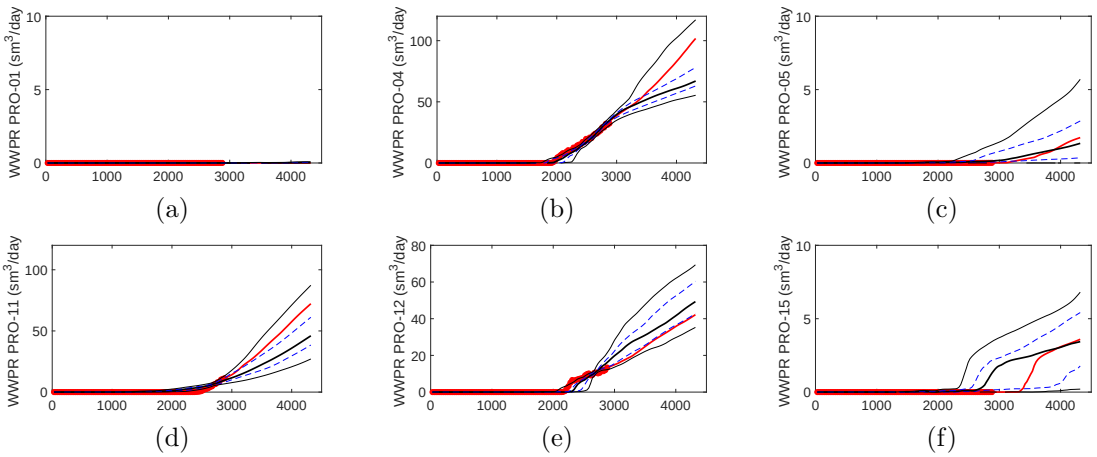


Figure 4.109: Marginal distribution for well water production rate for producers PRO-01 through PRO-15. In all figures, we show the corresponding true value (solid thick red curve), the mean of the marginal distribution (solid thick black curve), the percentiles P25 and P75 of the marginal distribution (two blue dashed curves), and the percentiles P5 and P95 of the marginal distribution (two thin black curves). To generate the marginal distributions presented here, all states after state 40,000 in the five chains are used, which gives a total of 300,000 states.

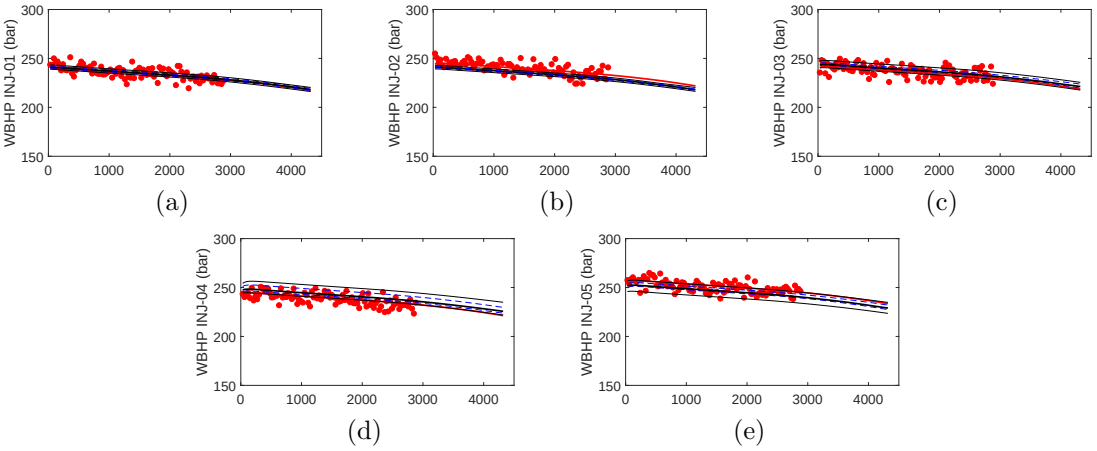


Figure 4.110: Marginal distribution for well bottom hole pressure for injectors INJ-01 through INJ-05. In all figures, we show the corresponding true value (solid thick red curve), the mean of the marginal distribution (solid thick black curve), the percentiles P25 and P75 of the marginal distribution (two blue dashed curves), and the percentiles P5 and P95 of the marginal distribution (two thin black curves). To generate the marginal distributions presented here, all states after state 40,000 in the five chains are used, which gives a total of 300,000 states.

CHAPTER 5

CONCLUSIONS

The methodology developed in this dissertation to apply the Metropolis-Hastings Markov chain Monte Carlo (MCMC) algorithm is a computationally efficient instrument for uncertainty quantification of reservoir model parameters and production forecasting. This methodology constructs the Markov chain using the predictions from a proxy model as a replacement of the reservoir simulator. As a consequence, no reservoir simulation run is required to construct a Markov chain when quantifying uncertainty with MCMC. Naturally, the proxy model runs far faster than the reservoir simulator, consequently, the computational burden of MCMC is significantly reduced.

Similar to Li and Reynolds [69], we construct a Gaussian mixture model (GMM) approximation of the posterior probability density function (pdf) to employ as the proposal distribution for the Metropolis-Hastings MCMC sampling algorithm. The GMM approximation is constructed centered at modes of the posterior pdf. The modes are found using a gradient-based minimization framework, but to alleviate the need of an adjoint solution, we instead use the analytical proxy gradient to conduct the minimization process. The history matching results we achieve using the proxy gradient corroborate that the proxy gradient is a suitable computational efficient alternative to the adjoint solution.

The proxy model investigated in this dissertation is the machine learning technique known as least-squares support vector regression (LS-SVR). The LS-SVR method constructs an approximate function using a given training set. The training examples which are included in the training set dictates the quality of the resulting LS-SVR model. To achieve a reliable proxy, capable of replacing the reservoir simulator, we construct the training set with an adaptive approach. For this, we dynamically update an initial training set by conducting

several minimization problems simultaneously. We conduct the minimization problems with the purpose of finding the modes of the posterior pdf to construct the GMM proposal distribution. By conducting all minimization problems simultaneously, we take advantage of a LS-SVR proxy gradient, which is dynamically updated at each minimization iteration. We update the training set by adding the estimates of the minima to a current training set at each iteration of the minimization problem. At the end of each iteration, an improved LS-SVR proxy model is trained using the current available training set, and the gradient of this proxy is used in the next minimization iteration. The underlying idea is that while the minimization proceeds, the estimates of the minima come from regions of increasing probability of the posterior pdf, consequently, the training set is adapted to regions of high probability. When conducting the MCMC sampling procedure, one is interested in regions of high probability, which are precisely the regions in which the LS-SVR proxy model constructed using the updated training set will provide accurate predictions. Hence, the LS-SVR proxy constructed becomes a suitable replacement of the reservoir simulator. The uncertainty quantification results presented in this dissertation corroborates this idea, with the advantage of considerably reducing the required number of reservoir simulator runs.

For large scale problems, we first apply principal component analysis (PCA) to the correlation matrix to reduce the dimension of a given problem, then we train the LS-SVR proxy model for the reduced-order input vector. A fortunate consequence of this approach is a further reduction in the computational burden, since the proxy constructed using the reduced-order vector runs faster than an equivalent LS-SVR proxy trained using the full dimensional input vector. Moreover, the MCMC sampling algorithm is faster, because we sample from a lower-dimensional multivariate Gaussian distribution and adapt a lower-dimensional covariance matrix. The computational efficiency and history matching and uncertainty quantification obtained when applying the developed methodology for a large scale reservoir model shows promising results, and should enable the application of the proposed methodology to real reservoir cases. We actually have assembled data for a large-scale field case, but do not have the computational resources to run the reservoir simulation model

several hundred to a few thousand times needed to build the training set.

5.1 Future Work

Motivated from the results presented in this dissertation, we intend to apply the developed methodology for a large-scale field case.

One open question for the machine learning support vector machines is the selection of the kernel function. Although the common recommendation is to use some available family of well-known kernel functions, which is natural for techniques developed for case where no clear physical model is available, we intend to investigate how one can use a given physical model to derive/improve a kernel function.

We also intend to investigate how one can couple the developed methodology with the ES-MDA method to further improve the quality of the uncertainty quantification, while reducing the computational burden.

BIBLIOGRAPHY

[1] S. I. Aannonsen, A. Cominelli, O. Gosselin, I. Aavatsmark, and T. Barkve. Integration of 4D data in the history match loop by investigating scale dependent correlations in the acoustic impedance cube. In *Proceedings of 8th European Conference on the Mathematics of Oil Recovery*, pages 1–8, Freiberg, Germany, 2002.

[2] Sigurd Ivar Aanonsen, Geir Nævdal, Dean S. Oliver, Albert C. Reynolds, and Brice Vallès. The ensemble Kalman filter in reservoir engineering–a review. *SPE Journal*, 14(03):393–412, 2009.

[3] Milton Abramowitz and Irene A. Stegun. *Handbook of Mathematical Functions: with Formulas, Graphs, and Mathematical Tables*, volume 55. Courier Corporation, 1965.

[4] M. A. Aizerman, E. M. Braverman, and L. I. Rozonoer. Theoretical foundation of potential functions method in pattern recognition [in Russian]. *Avtomatika i Telemekhanika*, 25(6):917–936, 1964.

[5] M. A. Aizerman, E. M. Braverman, and L. I. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning [Translated from [4]]. *Automation and Remote Control*, 25(6):821–837, 1964.

[6] Francois Alabert. The practice of fast conditional simulations through the LU decomposition of the covariance matrix. *Mathematical Geology*, 19(5):369–386, Jul 1987.

[7] Theodore W. Anderson and Raghu Raj Bahadur. Classification into two multivariate normal distributions with different covariance matrices. *The Annals of Mathematical Statistics*, 33(2):420–431, 1962.

[8] Christophe Andrieu and Éric Moulines. On the ergodicity properties of some adaptive MCMC algorithms. *The Annals of Applied Probability*, 16(3):1462–1505, 2006.

[9] Yves F. Atchadé and Jeffrey S. Rosenthal. On adaptive Markov chain Monte Carlo algorithms. *Bernoulli*, 11(5):815–828, 2005.

[10] Francis R. Bach. Consistency of the group lasso and multiple kernel learning. *Journal of Machine Learning Research*, 9(Jun):1179–1225, 2008.

[11] Francis R. Bach, Gert R. G. Lanckriet, and Michael I. Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *Proceedings of the Twenty-First International Conference on Machine Learning*, pages 41–48. ACM, 2004.

[12] Luciane Bonet-Cunha, Dean S. Oliver, Richard A. Redner, and Albert C. Reynolds. A hybrid Markov chain Monte Carlo method for generating permeability fields conditioned to multiwell pressure data and prior information. *SPE Journal*, 3(03):261–271, 1998. SPE–50991–PA.

[13] François Bouttier. A dynamical estimation of forecast error covariances in an assimilation system. *Monthly Weather Review*, 122(10):2376–2390, 1994.

[14] Stephen P. Brooks and Andrew Gelman. General methods for monitoring convergence of iterative simulations. *Journal of Computational and Graphical Statistics*, 7(4):434–455, 1998.

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

[16] David S. Broomhead and David Lowe. Radial basis functions, multi-variable functional interpolation and adaptive networks. Technical report, Royal Signals and Radar Establishment, Malvern – United Kingdom, 1988.

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

[18] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[19] G. Chavent, M. Dupuy, and P. Lemmonier. History matching by use of optimal theory. *Society of Petroleum Engineers Journal*, 15(01):74–86, Feb 1975.

[20] Wen H. Chen, George R. Gavalas, John H. Seinfeld, and Mel L. Wasserman. A new algorithm for automatic history matching. *Society of Petroleum Engineers Journal*, 14 (6):593–608, 1974.

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

[22] Lifu Chu, Albert C. Reynolds, and Dean S. Oliver. Computation of sensitivity coefficients for conditioning the permeability field to well-test pressure data. *In Situ*, 19(2): 179–223, 1995.

[23] Lifu Chu, Albert C. Reynolds, and Dean S. Oliver. Reservoir description from static and well-test data using efficient gradient methods. In *Proceedings of the International Meeting on Petroleum Engineering*, page 16 pages, Beijing, China, 14–17 November 1995. Society of Petroleum Engineers. SPE–29999–MS.

[24] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20 (3):273–297, 1995.

[25] Thomas M. Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Transactions on Electronic Computers*, EC-14(3):326–334, June 1965. ISSN 0367-7508. doi: 10.1109/PGEC.1965.264137.

[26] Michael W. Davis. Production of conditional simulations via the LU triangular decomposition of the covariance matrix. *Mathematical Geology*, 19(2):91–98, Feb 1987.

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

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

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

[30] 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 assimilation. In *Proccedings of the SPE Reservoir Simulation Symposium*, The Woodlands, Texas, USA, 18–20 February 2013. SPE–163675–MS.

[31] Geir Evensen. Using the extended Kalman filter with a multilayer quasi-geostrophic ocean model. *Journal of Geophysical Research: Oceans*, 97(C11):17905–17924, 1992.

[32] Geir Evensen. Open boundary conditions for the extended Kalman filter with a quasi-geostrophic ocean model. *Journal of Geophysical Research: Oceans*, 98(C9):16529–16546, 1993.

[33] 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.

[34] Geir Evensen. *Data assimilation: the ensemble Kalman filter*. Springer Science & Business Media, New York, New York, USA, second edition, 2009.

[35] William Feller. *An Introduction to Probability Theory and Its Applications*, volume 1 of *Wiley Series in Probability and Mathematical Statistics*. John Wiley & Sons, New York, New York, USA, third edition, 1968.

[36] F. J. T. Floris, M. D. Bush, M. Cuypers, F. Roggero, and A. R. Syversveen. Methods for quantifying the uncertainty of production forecasts: a comparative study. *Petroleum Geoscience*, 7(S):S87–S96, 2001.

[37] Guohua Gao. *Data Integration and Uncertainty Evaluation for Large Scale Automatic History Matching*. Ph.D. dissertation, The University of Tulsa, Tulsa, OK, 2005.

[38] Guohua Gao, Mohammad Zafari, and Albert C. Reynolds. Quantifying uncertainty for the PUNQ-S3 problem in a Bayesian setting with RML and EnKF. *SPE Journal*, 11(04):506–515, 2006. SPE–93324–PA.

[39] Guohua Gao, Jeroen C. Vink, Chaohui Chen, Mohammadali Tarrahi, and Yaakoub El Khamra. Uncertainty quantification for history matching problems with multiple best matches using a distributed Gauss-Newton method. In *SPE Annual Technical Conference and Exhibition*. Society of Petroleum Engineers, 2016.

[40] Guohua Gao, Hao Jiang, Paul van Hagen, Jeroen C. Vink, and Terence Wells. A Gauss-Newton trust-region solver for large-scale history-matching problems. *SPE Journal*, 22(06):1999–2011, Dec 2017.

[41] 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*, 21(5):1325–1342, Dec 2017.

[42] Pierre Gauthier, Philippe Courtier, and Patrick Moll. Assimilation of simulated wind lidar data with a Kalman filter. *Monthly Weather Review*, 121(6):1803–1820, 1993.

[43] A. Gelman, G. O. Roberts, and W. R. Gilks. Efficient Metropolis jumping rules. *Bayesian Statistics. Oxford University Press*, 5(5):599–607, 1996.

[44] W. R. Gilks, S. Richardson, and D. J. Spiegelhalter, editors. *Markov Chain Monte Carlo in Practice*. Chapman & Hall, New York, New York, USA, 1996.

[45] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, Maryland, USA, third edition, 1996.

[46] Gene H. Golub and Christian Reinsch. Singular value decomposition and least squares solutions. *Numerische mathematik*, 14(5):403–420, 1970.

[47] Mehmet Gönen and Ethem Alpaydın. Multiple kernel learning algorithms. *Journal of Machine Learning Research*, 12(Jul):2211–2268, 2011.

[48] Nicholas I. M. Gould, Daniel P. Robinson, and H. Sue Thorne. On solving trust-region and other regularised subproblems in optimization. *Mathematical Programming Computation*, 2(1):21–57, 2010.

[49] Yaqing Gu and Dean S. Oliver. An iterative ensemble Kalman filter for multiphase fluid flow data assimilation. *SPE Journal*, 12(04):438–446, 2007. SPE–108438–PA.

[50] Heikki Haario, Eero Saksman, and Johanna Tamminen. Adaptive proposal distribution for random walk Metropolis algorithm. *Computational Statistics*, 14(3):375–396, 1999.

[51] Heikki Haario, Eero Saksman, and Johanna Tamminen. An adaptive Metropolis algorithm. *Bernoulli*, 7(2):223–242, 2001.

[52] Martin Hanke. A regularizing Levenberg-Marquardt scheme, with applications to inverse groundwater filtration problems. *Inverse Problems*, 13(1):79–95, 1997.

[53] Nikolaus Hansen. The cma evolution strategy: a comparing review. In *Towards a New Evolutionary Computation*, pages 75–102. Springer, 2006.

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

[55] Per Christian Hansen. The truncated SVD as a method for regularization. *BIT Numerical Mathematics*, 27(4):534–553, 1987.

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

[57] Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[58] Lars Holden, Ragnar Hauge, and Marit Holden. Adaptive independent Metropolis–Hastings. *The Annals of Applied Probability*, 19(1):395–413, 2009.

[59] Peter L. Houtekamer and Herschel L. Mitchell. Data assimilation using an ensemble Kalman filter technique. *Monthly Weather Review*, 126(3):796–811, 1998.

[60] Marco A. Iglesias. Iterative regularization for ensemble-based data assimilation in reservoir models. *Computational Geosciences*, 19(1):177–212, 2015.

[61] Marco A. Iglesias and Clint Dawson. The regularizing Levenberg–Marquardt scheme for history matching of petroleum reservoirs. *Computational Geosciences*, 17(6):1033–1053, 2013.

[62] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

[63] Leonard Kaufman and Peter J. Rousseeuw. *Finding Groups in Data: an Introduction to Cluster Analysis*, volume 59 of *Wiley Series in Probability and Statistics*. John Wiley & Sons, Hoboken, New Jersey, USA, 2009.

[64] Peter K. Kitanidis. Quasi-linear geostatistical theory for inversing. *Water Resources Research*, 31(10):2411–2419, 1995.

[65] 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(06):2195–2207, 2016. SPE–173214–PA.

[66] Kenneth Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, 2(2):164–168, 1944.

[67] R. Li, Albert C. Reynolds, and Dean S. Oliver. Sensitivity coefficients for three-phase flow history matching. *Journal of Canadian Petroleum Technology*, 42(04):70–77, 2003.

[68] Xin Li. *Development of a Proposal Distribution for an Efficient Metropolis-Hastings MCMC Algorithm for the Characterization of Uncertainty in Reservoir Description and Production Forecasts*. Ph.D. dissertation, The University of Tulsa, Tulsa, Oklahoma, USA, 2017.

[69] Xin Li and Albert C. Reynolds. Generation of a proposal distribution for efficient MCMC characterization of uncertainty in reservoir description and forecasting. In *Proceedings of the SPE Reservoir Simulation Conference*, Montgomery, Texas, USA, 20–22 February 2017. Society of Petroleum Engineers. SPE–182684–MS.

[70] Faming Liang, Chuanhai Liu, and Raymond Carroll. *Advanced Markov chain Monte Carlo Methods: Learning from Past Samples*, volume 714 of *Wiley Series in Computational Statistics*. John Wiley & Sons, 2010.

[71] Ning Liu and Dean S. Oliver. Evaluation of Monte Carlo methods for assessing uncertainty. *SPE Journal*, 8(02):188–195, 2003.

[72] Xiaodong Luo, Andreas S. Stordal, Rolf J. Lorentzen, and Geir Nævdal. Iterative ensemble smoother as an approximate solution to a regularized minimum-average-cost problem: theory and applications. *SPE Journal*, 20(05):962–982, 2015. SPE–176023–PA.

[73] Donald W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[74] Michael D. McKay, Richard J. Beckman, and William J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 42(1):55–61, 2000.

[75] J. Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 209(441-458):415–446, 1909.

[76] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[77] Robert N. Miller, Michael Ghil, and Francois Gauthiez. Advanced data assimilation in strongly nonlinear dynamical systems. *Journal of the Atmospheric Sciences*, 51(8): 1037–1056, 1994.

[78] Eliakim H. Moore. On the reciprocal of the general algebraic matrix. *Bulletin of the American Mathematical Society*, 26(9):394–395, 1920.

[79] Jorge J. Moré. The levenberg-marquardt algorithm: Implementation and theory. In G. A. Watson, editor, *Numerical Analysis*, pages 105–116, Berlin, Germany, 1978. Springer.

[80] Jorge J. Moré and Danny C. Sorensen. Computing a trust region step. *SIAM Journal on Scientific and Statistical Computing*, 4(3):553–572, Sep 1983.

[81] 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*, Tulsa, Oklahoma, USA, 13–17 April 2002. SPE–75235–MS.

[82] Geir Nævdal, Liv Merete Johnsen, Sigurd Ivar Aanonsen, and Erlend H. Vefring. Reservoir monitoring and continuous model updating using ensemble Kalman filter. *SPE Journal*, 10(01):66–74, 2005. SPE–84372–PA.

[83] J. Nocedal and S. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, New York, New York, USA, second edition, 2006. ISBN 9780387400655. URL https://books.google.com/books?id=VbHYoSyelFcC.

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

[85] Dean S. Oliver. Metropolized randomized maximum likelihood for improved sampling from multimodal distributions. *SIAM/ASA Journal on Uncertainty Quantification*, 5 (1):259–277, 2017.

[86] Dean S Oliver and Yan Chen. Recent progress on reservoir history matching: a review. *Computational Geosciences*, 15(1):185–221, 2011.

[87] 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.

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

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

[90] Roger Penrose. A generalized inverse for matrices. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 51, pages 406–413. Cambridge University Press, 1955.

[91] Alvaro M. M. Peres. *Analysis of Slug and Drillstem Tests*. Ph.D. dissertation, The University of Tulsa, Tulsa, Oklahoma, USA, 1989.

[92] Fernando Pérez-Cruz and Olivier Bousquet. Kernel methods and their potential use in signal processing. *IEEE Signal Processing Magazine*, 21(3):57–65, May 2004. ISSN 1053-5888. doi: 10.1109/MSP.2004.1296543.

[93] Javad Rafiee. *Data Assimilation and Uncertainty Quantification With Ensemble Methods and Markov Chain Monte Carlo*. Ph.D. dissertation, The University of Tulsa, Tulsa, Oklahoma, USA, 2017.

[94] 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.

[95] Javad Rafiee and Albert C. Reynolds. A two-level MCMC based on the distributed Gauss-Newton method for uncertainty quantification. In *ECMOR XVI-16th European Conference on the Mathematics of Oil Recovery*, Barcelona, Spain, 3–6 September 2018.

[96] Albert C. Reynolds, Nanqun He, Lifu Chu, and Dean S. Oliver. Reparameterization techniques for generating reservoir descriptions conditioned to variograms and well-test pressure data. *SPE Journal*, 1(04):413–426, 1996. SPE-30588-PA.

[97] 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.

[98] Albert C. Reynolds, R. Li, and Dean S. Oliver. Simultaneous estimation of absolute and relative permeability by automatic history matching of three-phase flow production data. *Journal of Canadian Petroleum Technology*, 43(03):37–46, 2004.

[99] Albert C. Reynolds, Mohammad Zafari, and Gaoming Li. Iterative forms of the ensemble kalman filter. In *ECMOR X-10th European Conference on the Mathematics of Oil Recovery*, Amsterdam, The Netherlands, 4–7 September 2006.

[100] Douglas Reynolds. Gaussian mixture models. In Stan Z. Li and Anil K. Jain, editors, *Encyclopedia of Biometrics*, pages 827–832. Springer, Boston, Massachusetts, US, 2015.

[101] G. O. Roberts and J. S. Rosenthal. Optimal scaling of various Metropolis-Hastings algorithms. *Statistical Science*, 16(4):351–367, 2001.

[102] 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.

[103] Gareth O. Roberts and Jeffrey S. Rosenthal. Coupling and ergodicity of adaptive Markov chain Monte Carlo algorithms. *Journal of applied probability*, 44(2):458–475, 2007.

[104] Gareth O. Roberts and Richard L. Tweedie. Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, 2(4):341–363, 1996.

[105] Joris Rombout Rommelse. *Data Assimilation in Reservoir Management*. Ph.D. dissertation, Technische Universiteit Delft - Technical University of Delft, Delft, The Netherlands, 2009.

[106] Jeffrey S. Rosenthal. Optimal proposal distributions and adaptive MCMC [Chapter for [15]]. 2011.

[107] Arthur L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, 1959.

[108] Bernhard Schölkopf, Patrice Simard, Alex J. Smola, and Vladimir Vapnik. Prior knowledge in support vector kernels. In *Advances in Neural Information Processing Systems*, pages 640–646, 1998.

[109] Alexandra Seiler, Sigurd Ivar Aanonsen, Geir Evensen, and Jan C. Reivenæs. Structural uncertainty modeling and updating using the ensemble Kalman filter. *SPE Journal*, 15(4):1062–1076, 2010. SPE–125352–PA.

[110] Jonathon Shlens. A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100*, 2014.

[111] Jan-Arild Skjervheim, Geir Evensen, Sigurd Ivar Aanonsen, Bent Ole Ruud, and Tor-Arne Johansen. Incorporating 4D seismic data in reservoir simulation models using ensemble Kalman filter. *SPE Journal*, 12(03):282–292, 2007. SPE–95789–PA.

[112] Alex J. Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, 2004.

[113] Danny C. Sorensen. Newton's method with a model trust region modification. *SIAM Journal on Numerical Analysis*, 19(2):409–426, Apr 1982.

[114] Harald Stehfest. Algorithm 368: Numerical inversion of Laplace transforms [D5]. *Communications of the ACM*, 13(1):47–49, 1970.

[115] Ingo Steinwart. On the influence of the kernel on the consistency of support vector machines. *Journal of Machine Learning Research*, 2(Nov):67–93, 2001.

[116] Andreas S. Stordal and Geir Nævdal. A modified randomized maximum likelihood for improved Bayesian history matching. *Computational Geosciences*, 22(1):29–41, 2018.

[117] Johan A. K. Suykens. Least squares support vector machines for classification and nonlinear modelling. *Neural Network World*, 10(1-2):29–48, 2000.

[118] Johan A. K. Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, 1999.

[119] Johan A. K. Suykens and Joos Vandewalle. Recurrent least squares support vector machines. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 47(7):1109–1114, 2000.

[120] Johan A. K. Suykens, Jos De Brabanter, Lukas Lukas, and Joos Vandewalle. Weighted least squares support vector machines: robustness and sparse approximation. *Neurocomputing*, 48(1-4):85–105, 2002.

[121] Johan A. K. Suykens, Tony Van Gestel, Jos De Brabanter, Bart De Moor, and Joos Vandewalle. *Least Squares Support Vector Machines*. World Scientific, 2002.

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

[123] Reza Tavakoli and Albert C. Reynolds. Monte Carlo simulation of permeability fields and reservoir performance predictions with SVD parameterization in RML compared with EnKF. *Computational Geosciences*, 15(1):99–116, 2011.

[124] 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, USA, 11–14 November 2007. SPE–109975–MS.

[125] Luke Tierney. Markov chains for exploring posterior distributions. *The Annals of Statistics*, 22(4):1701–1728, 1994.

[126] Peter Jan van Leeuwen and Geir Evensen. Data assimilation and inverse methods in terms of a probabilistic formulation. *Monthly Weather Review*, 124(12):2898–2913, 1996.

[127] Vladimir N. Vapnik. *Estimation of Dependences Based on Empirical Data* [in Russian]. Nauka, Moscow, Russia, 1979.

[128] Vladimir N. Vapnik. *Estimation of Dependences Based on Empirical Data* [Translated from [127]]. Springer Series in Statistics. Springer-Verlag, 1982. ISBN 0-387-90733-5.

[129] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, New York, USA, 1995. ISBN 0-387-94559-8.

[130] Vladimir N. Vapnik. *Statistical Learning Theory*. Adaptive and Learning Systems for Signal Processing, Communications, and Control. John Wiley & Sons, 1998. ISBN 9780471030034.

[131] Vladimir N. Vapnik and A. Ya Chervonenkis. On a perceptron class [in Russian]. *Avtomatika i Telemekhanika*, 25(1):112–120, 1964.

[132] Vladimir N. Vapnik and A. Ya Chervonenkis. On a class of perceptrons [Translated from [131]]. *Automation and Remote Control*, 25(1):103–109, 1964.

[133] Vladimir N. Vapnik and A. Ya Chervonenkis. *Theory of Pattern Recognition: Statistical Problems of Learning* [in Russian]. Nauka, Moscow, Russia, 1974.

[134] Vladimir N. Vapnik and A. Ya Lerner. Recognition of patterns with help of generalized portraits [in Russian]. *Avtomatika i Telemekhanika*, 24(6):774–780, 1963.

[135] Vladimir N. Vapnik and A. Ya Lerner. Pattern recognition using generalized portraits [Translated from [134]]. *Automation and Remote Control*, 24(6):709–715, 1963.

[136] Yudou Wang, Gaoming Li, and Albert C. Reynolds. Estimation of depths of fluid contacts by history matching using iterative ensemble-Kalman smoothers. *SPE Journal*, 15(2), 2010. SPE–119056–PA.

[137] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 2(1–3):37–52, 1987.

[138] Guoshen Yu, Guillermo Sapiro, and Stéphane Mallat. Solving inverse problems with piecewise linear estimators: From Gaussian mixture models to structured sparsity. *IEEE Transactions on Image Processing*, 21(5):2481–2499, 2012.

[139] Mohammad Zafari and Albert C. Reynolds. Assessing the uncertainty in reservoir description and performance predictions with the ensemble Kalman filter. *SPE Journal*, 12(03):382–391, 2007. SPE–95750–PA.

[140] Fengjun Zhang, Albert C. Reynolds, and Dean S. Oliver. Evaluation of the reduction in uncertainty obtained by conditioning a 3D stochastic channel to multiwell pressure data. *Mathematical Geology*, 34(6):715–742, 2002.

[141] Yong Zhao, Gaoming Li, and Albert C. Reynolds. Characterization of the measurement error in time-lapse seismic data and production data with an EM algorithm. *Oil & Gas Science and Technology - Revue de l'IFP*, 62(2):181–193, 2007.

APPENDIX A

TRAINING PROCEDURE FOR THE LEAST SQUARES SUPPORT VECTOR REGRESSION

To construct a least-squares support vector regression (LS-SVR) approximation for some nonlinear function $f(\boldsymbol{x})$, which acts on a N_x-dimensional column vector of input parameters, $\boldsymbol{x}$, to generate the N_y-dimensional column vector of output data, $\boldsymbol{y}$, i.e.,

$$\boldsymbol{y} = f(\boldsymbol{x}) \,, \tag{A.1}$$

the LS-SVR regression method requires a training set. A training set is composed by a suite of vectors of input parameters and their respective vectors of output data which are related by Eq. A.1.

The LS-SVR method proceeds by constructing an ensemble of N_y regression functions, denoted by $\hat{f}_i(\boldsymbol{x})$, for $i = 1, 2, \ldots, N_y$. Note we have one independent regression function $\hat{f}_i(\boldsymbol{x})$ for each entry y_i, for $i = 1, 2, \ldots, N_y$, of the column vector of output data $\boldsymbol{y}$. The set of N_y regression functions $\hat{f}(\boldsymbol{x}) = [\hat{f}_1(\boldsymbol{x}), \hat{f}_2(\boldsymbol{x}), \ldots, \hat{f}_{N_y}(\boldsymbol{x})]^T$ emulates the original function $f(\boldsymbol{x})$ of Eq. A.1 when computing outputs for input vectors not present in the original training set. A particular set o regression functions $\hat{f}(\boldsymbol{x})$, constructed using a given training set, is what we collectively denote as a LS-SVR proxy model approximation for the original function $f(\boldsymbol{x})$ of Eq. A.1.

As proposed by Vapnik [129, 130], the nonlinear LS-SVR method adopted in this research simply applies a linear LS-SVR on some high-dimensional feature space. Hence, the LS-SVR algorithm for nonlinear functions starts by first mapping the input vectors $\boldsymbol{x}$, of a

given training set, from its original input space into a high-dimensional feature space using a chosen nonlinear mapping. One can define the resulting feature space by a particular choice of the nonlinear mapping. However, the selected mapping does not need to be explicitly defined, as we discuss later.

The underlying key idea of the nonlinear LS-SVR method is to choose a nonlinear mapping which defines a feature space where the relationship between the mapped input vectors and their respective outputs becomes linear. Considering that a particular nonlinear mapping $\varphi_i(\boldsymbol{x})$ defines a feature space where the relationship between $\boldsymbol{x}$ and y_i, where y_i denotes the ith entry of the vector of outputs, $\boldsymbol{y}$, for $i = 1, 2, \ldots, N_y$, becomes linear, one searches for a linear relationship on the feature space in the form

$$\hat{f}_i(\boldsymbol{x}) = \boldsymbol{\omega}_i^T \varphi_i(\boldsymbol{x}) + b_i \qquad \text{for} \quad i = 1, 2, \ldots, N_y \,, \tag{A.2}$$

where $\boldsymbol{\omega}_i$ represents the column vector of coefficients of $\varphi_i(\boldsymbol{x})$, and b_i is the bias term. Hence, the LS-SVR training algorithm effectively reduces to determine the coefficients $\boldsymbol{\omega}_i$ and b_i of Eq. A.2.

The dimension of the vector $\boldsymbol{\omega}_i$ depends on the chosen feature space. For some choices of feature spaces, $\boldsymbol{\omega}_i$ may be infinite dimensional. This could be a challenger for the training procedure, however $\boldsymbol{\omega}_i$ is not explicitly determined as shown below.

A.1 Determining the Least Squares Support Vector Regression Coefficients

In this research, we focus on the LS-SVR methodology proposed by [119] and [117], in which a least squares loss function is used as regularization term in a minimization problem designed to determining the coefficients $\boldsymbol{\omega}_i$ and b_i of Eq. A.2.

Given a particular training set $T_s = \{(\boldsymbol{x}_k, \boldsymbol{y}_k), \text{ for } k = 1, 2, \ldots, N_t\}$, where $\boldsymbol{x}_k$ and $\boldsymbol{y}_k$ are related by Eq. A.1, i.e., $\boldsymbol{y}_k = f(\boldsymbol{x}_k)$, one needs to solve the N_y minimization problems

[120, 121] given by

$$\underset{\boldsymbol{\omega}_i, b_i, \boldsymbol{e}_i}{\text{minimize}}\; J_i(\boldsymbol{\omega}_i, \boldsymbol{e}_i \mid \gamma_i) = \frac{1}{2}\boldsymbol{\omega}_i^T\boldsymbol{\omega}_i + \frac{1}{2}\gamma_i \sum_{k=1}^{N_t} e_{i,k}^2 \qquad \text{for} \quad i = 1,\, 2,\, \ldots\,,\, N_y\,, \tag{A.3}$$

subject to

$$e_{i,k} = y_{k,i} - \boldsymbol{\omega}_i^T \varphi_i(\boldsymbol{x}_k) - b_i\,, \qquad \text{for} \quad k = 1,\, 2,\, \ldots\,,\, N_t\,. \tag{A.4}$$

In Eqs. A.3 and A.4, $y_{k,i}$ represents the ith entry of the kth column vector of output data $\boldsymbol{y}_k = f(\boldsymbol{x}_k)$ from the given training set T_s, and $\boldsymbol{e}_i = [e_{i,k}]_{k=1}^{N_t} \equiv [e_{i,1},\, e_{i,2},\, \ldots\,,\, e_{i,N_t}]^T$ denotes the ith column vector of mismatch between the true output value $y_{k,i}$ and its respective prediction $\hat{f}_i(\boldsymbol{x}_k)$ given by Eq. A.2, i.e., the kth entry of the vector $\boldsymbol{e}_i$, for $i = 1,\, 2,\, \ldots\,,\, N_y$, is given by

$$e_{i,k} = y_{k,i} - \hat{f}_i(\boldsymbol{x}_k)\,, \qquad \text{for} \quad k = 1,\, 2,\, \ldots\,,\, N_t\,. \tag{A.5}$$

The second term in the right hand side of Eq. A.3 represents the least-squares feature of the algorithm, as mentioned earlier. The positive parameter γ_i controls the trade off between the flatness of the function $\hat{f}_i(\boldsymbol{x})$ and how much deviation between $y_{i,k}$ and $\hat{f}_i(\boldsymbol{x}_k)$ is allowed [112]. The parameter γ_i has to be determined before the training procedure, which could be considered a drawback of the method. However, as discussed in the main text of this dissertation, numerical experiments reveal that for values of $\gamma_i > 200$, the training procedure is nearly insensitive to the parameter γ_i. In this research, we adopt a constant value of $\gamma_i \equiv \gamma = 800$, for $i = 1,\, 2,\, \ldots\,,\, N_y$, in all training procedures which are performed.

The set of N_y minimization problems given by Eqs. A.3 and A.4, for $i = 1,\, 2,\, \ldots\,,\, N_y$, are solved using the method of Lagrange multipliers, see Nocedal and Wright [83]. The Lagrangian is constructed from Eqs. A.3 and A.4, which leads to

$$\mathcal{L}_i(\boldsymbol{\omega}_i, b_i, \boldsymbol{e}_i; \boldsymbol{\alpha}_i) = \frac{1}{2}\boldsymbol{\omega}_i^T\boldsymbol{\omega}_i + \frac{1}{2}\gamma_i \sum_{k=1}^{N_t} e_{i,k}^2 - \sum_{k=1}^{N_t} \alpha_{i,k}\left[e_{i,k} - y_{k,i} + \boldsymbol{\omega}_i^T\varphi_i(\boldsymbol{x}_k) + b_i\right],$$

$$\text{for} \quad i = 1,\, 2,\, \ldots\,,\, N_y\,, \tag{A.6}$$

where $\boldsymbol{\alpha}_i = [\alpha_{i,k}]_{k=1}^{N_t} \equiv [\alpha_{i,1},\, \alpha_{i,2},\, \ldots\, ,\, \alpha_{i,N_t}]^T$ denotes the vector of Lagrange multipliers for the ith minimization problem.

An optimal solution of the minimization problems defined by Eqs. A.3 and A.4 needs to satisfy the Karush–Kuhn–Tucker (KKT) conditions [83]. Hence, at an optimal solution we must have $\nabla \mathcal{L}_i = 0$ [83], which, from Eq. A.6, for $i = 1,\, 2,\, \ldots\, ,\, N_y$, implies that

$$\nabla_{\boldsymbol{\omega}_i} \mathcal{L}_i \;=\; 0 \quad \rightarrow \quad \boldsymbol{\omega}_i = \sum_{k=1}^{N_t} \alpha_{i,k}\, \varphi_i(\boldsymbol{x}_k)\,, \tag{A.7a}$$

$$\frac{\partial \mathcal{L}_i}{\partial b_i} \;=\; 0 \quad \rightarrow \quad \sum_{k=1}^{N_t} \alpha_{i,k} = 0\,, \tag{A.7b}$$

$$\frac{\partial \mathcal{L}_i}{\partial e_{i,k}} \;=\; 0 \quad \rightarrow \quad \alpha_{i,k} = \gamma_i\, e_{i,k}\,, \qquad \text{for} \quad k = 1,\, 2,\, \ldots\, ,\, N_t\,, \tag{A.7c}$$

$$\frac{\partial \mathcal{L}_i}{\partial \alpha_{i,k}} \;=\; 0 \quad \rightarrow \quad e_{i,k} - y_{k,i} + \boldsymbol{\omega}_i^T \varphi_i(\boldsymbol{x}_k) + b_i = 0\,, \qquad \text{for} \quad k = 1,\, 2,\, \ldots\, ,\, N_t\,. \tag{A.7d}$$

Substituting Eq. A.7a into Eq. A.7d, for $i = 1,\, 2,\, \ldots\, ,\, N_y$, one gets

$$y_{k,i} = \sum_{\ell=1}^{N_t} \alpha_{i,\ell}\, \varphi_i(\boldsymbol{x}_\ell)^T \varphi_i(\boldsymbol{x}_k) + b_i + e_{i,k}\,, \qquad \text{for} \quad k = 1,\, 2,\, \ldots\, ,\, N_t\,. \tag{A.8}$$

For $i = 1,\, 2,\, \ldots\, ,\, N_y$, substituting Eq. A.7c into Eq. A.8, leads to

$$y_{k,i} = \sum_{\ell=1}^{N_t} \alpha_{i,\ell}\, \varphi_i(\boldsymbol{x}_\ell)^T \varphi_i(\boldsymbol{x}_k) + b_i + \frac{\alpha_{i,k}}{\gamma_i}\,, \qquad \text{for} \quad k = 1,\, 2,\, \ldots\, ,\, N_t\,. \tag{A.9}$$

The set of N_t equations in Eq. A.9, for $i = 1,\, 2,\, \ldots\, ,\, N_y$, can be rewritten in matrix-vector form as

$$b_i\, \mathbf{1}_{N_t} + \left(\Omega_i + \frac{1}{\gamma_i} I_{N_t} \right) \boldsymbol{\alpha}_i = \boldsymbol{Y}_i\,. \tag{A.10}$$

In Eq. A.10, $\mathbf{1}_{N_t} = [1,\, 1,\, \ldots\, ,\, 1]_{N_t}^T$ denotes a N_t-dimensional column vector with all components equal to unity; I_{N_t} represents the $N_t \times N_t$ identity matrix; Ω_i denotes the $N_t \times N_t$ matrix with entry in the kth row and ℓth column given by the inner product $\Omega_{i,k,\ell} = \varphi_i(\boldsymbol{x}_k)^T \varphi_i(\boldsymbol{x}_\ell)$; $\boldsymbol{\alpha}_i = [\alpha_{i,1},\, \alpha_{i,2},\, \ldots\, ,\, \alpha_{i,N_t}]^T$ again represents the column vector of Lagrange multipliers for

the ith minimization problem; and $\boldsymbol{Y}_i = [y_{k,i}]_{k=1}^{N_t} \equiv [y_{1,i},\, y_{2,i},\, \ldots\, ,\, y_{N_t,i}]^T$ denotes the column vector with kth entry equal to the ith entry of the kth vector of output data, $\boldsymbol{y}_k$, from the training set T_s.

Combining Eqs. A.7b and A.10, for $i = 1, 2, \ldots, N_y$, one gets the following linear system

$$\begin{bmatrix} 0 & \mathbf{1}_{N_t}^T \\ \mathbf{1}_{N_t} & \Omega_i + \frac{1}{\gamma_i} I_{N_t} \end{bmatrix} \begin{bmatrix} b_i \\ \boldsymbol{\alpha}_i \end{bmatrix} = \begin{bmatrix} 0 \\ \boldsymbol{Y}_i \end{bmatrix} , \tag{A.11}$$

which the solution provides

$$b_i = \frac{\mathbf{1}_{N_t}^T \left(\Omega_i + \frac{1}{\gamma_i} I_{N_t}\right)^{-1} \boldsymbol{Y}_i}{\mathbf{1}_{N_t}^T \left(\Omega_i + \frac{1}{\gamma_i} I_{N_t}\right)^{-1} \mathbf{1}_{N_t}} , \tag{A.12}$$

and

$$\boldsymbol{\alpha}_i = \left(\Omega_i + \frac{1}{\gamma_i} I_{N_t}\right)^{-1} \boldsymbol{Y}_i \, - \, b_i \left(\Omega_i + \frac{1}{\gamma_i} I_{N_t}\right)^{-1} \mathbf{1}_{N_t} . \tag{A.13}$$

For the ith LS-SVR regression function $\hat{f}_i(\boldsymbol{x})$ of Eq. A.2, for $i = 1, 2, \ldots, N_y$, the coefficient b_i is given by Eq. A.12, while the coefficient $\boldsymbol{\omega}_i$ is implicitly determined from Eq. A.7a using the Lagrange multiplier $\boldsymbol{\alpha}_i$ given by Eq. A.13. Thus, there is no need to explicitly compute $\boldsymbol{\omega}_i$. This is a key result because $\boldsymbol{\omega}_i$ could be infinite dimensional for some choices of the mapping $\varphi_i(\boldsymbol{x})$.

Note that to compute Eqs. A.12 and A.13, one needs to solve a $N_t \times N_t$ matrix problem. To solve the $N_t \times N_t$ matrix problem of Eqs. A.12 and A.13, we use the conjugated gradient method [83].

A.2 The Kernel Trick

Finally, substituting Eq. A.7a into Eq. A.2, for $i = 1, 2, \ldots, N_y$, we obtain

$$\hat{f}_i(\boldsymbol{x}) = \sum_{k=1}^{N_t} \alpha_{i,k}\, \varphi_i(\boldsymbol{x}_k)^T \varphi_i(\boldsymbol{x}) + b_i \, , \tag{A.14}$$

where b_i is obtained from Eq. A.12 and $\alpha_{i,k}$ is the kth entry of the vector $\boldsymbol{\alpha}_i$ from Eq. A.13.

To compute Eq. A.14 for a given vector $\boldsymbol{x}$, one only needs inner products between the mapped vectors $\varphi_i(\boldsymbol{x})$ and $\varphi_i(\boldsymbol{x}_k)$. It may appear that one needs to specify the mapping $\varphi_i(\boldsymbol{x})$ before computing such an inner products. However, one can select a kernel function to provide the required inner products, i.e., given two vectors $\boldsymbol{x}_r$ and $\boldsymbol{x}_s$, we define

$$K_i(\boldsymbol{x}_r, \boldsymbol{x}_s) = \varphi_i(\boldsymbol{x}_r)^T \varphi_i(\boldsymbol{x}_s)\,, \qquad \text{for} \quad i = 1,\, 2,\, \ldots\,,\, N_y\,. \tag{A.15}$$

Therefore, it is not necessary to explicitly compute the mapping $\varphi_i(\boldsymbol{x})$. That is the main advantage of any support vector method. Replacing the inner products in the feature space by some kernel function became known as the kernel trick. In practice, the kernel function $K_i(\boldsymbol{x}_r, \boldsymbol{x}_s)$ of Eq. A.15 is defined as a function of the vectors $\boldsymbol{x}_r$ and $\boldsymbol{x}_s$ only. It is believed that, for any nonlinear function, one can always determine a kernel function that defines a feature space in which the relationship between input and output vectors becomes linear, see Pérez-Cruz and Bousquet [92], for example. However, a general clear procedure of how to derive such a kernel function is yet to be discovered, see Smola and Schölkopf [112], for example. The selection of a determined kernel function for a specific given problem is discussed in the main text of this dissertation.

Using the kernel trick of Eq. A.15, Eq. A.14 reduces to

$$\hat{f}_i(\boldsymbol{x}) = \sum_{k=1}^{N_t} \alpha_{i,k}\, K_i(\boldsymbol{x}_k, \boldsymbol{x}) + b_i \qquad \text{for} \quad i = 1,\, 2,\, \ldots\,,\, N_y\,, \tag{A.16}$$

which is used to compute $\hat{f}_i(\boldsymbol{x})$ for any given vector $\boldsymbol{x}$.

APPENDIX B

TRUST REGION MINIMIZATION ALGORITHM

In this Appendix, we present a trust region algorithm to minimizing a particular real-valued function, $O(\boldsymbol{x})$, of the form

$$O(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{x} + \frac{1}{2}\boldsymbol{y}(\boldsymbol{x})^T\boldsymbol{y}(\boldsymbol{x})\,, \tag{B.1}$$

for which the gradient, $\mathcal{G}(\boldsymbol{x})$, and Hessian, $\mathcal{H}(\boldsymbol{x})$, are given respectively by

$$\mathcal{G}(\boldsymbol{x}) \equiv \nabla_{\boldsymbol{x}}\, O(\boldsymbol{x}) = \boldsymbol{x} + G(\boldsymbol{x})^T\boldsymbol{y}(\boldsymbol{x}) \tag{B.2}$$

and

$$\mathcal{H}(\boldsymbol{x}) = I_{N_x} + G(\boldsymbol{x})^T\, G(\boldsymbol{x})\,. \tag{B.3}$$

In Eqs. B.1 through B.3, $\boldsymbol{x}$ represents a N_x-dimensional column vector of inputs, $\boldsymbol{y}$ represents a N_y-dimensional column vector of outputs, which is assumed to depend on some nonlinear functional of $\boldsymbol{x}$, and I_{N_x} denotes the $N_x \times N_x$ identity matrix. The $N_y \times N_x$ matrix $G(\boldsymbol{x})$ denotes a sensitivity matrix, i.e.,

$$G(\boldsymbol{x}) = \left[\nabla_{\boldsymbol{x}}\left(\boldsymbol{y}(\boldsymbol{x})^T\right)\right]^T\,. \tag{B.4}$$

Hence, the entry in the ith row and jth column of the sensitivity matrix $G(\boldsymbol{x})$ represents the partial derivative of the ith entry of the output vector $\boldsymbol{y}$ with respect to the jth entry of the input vector $\boldsymbol{x}$.

The Gauss-Newton trust region minimization algorithm presented here was developed

by Rafiee [93] and is based on the work of Gao et al. [40]. The trust region algorithm proceeds by minimizing a quadratic approximation of the function $O(\boldsymbol{x})$ of Eq. B.1 in a hyper-sphere region centered at the current estimate of the minimum of $O(\boldsymbol{x})$, as follows

$$\underset{\delta\boldsymbol{x}}{\text{minimize}} \;\; q_p^{(n)}(\delta\boldsymbol{x}) = O(\boldsymbol{x}_p^{(n)}) + \mathcal{G}(\boldsymbol{x}_p^{(n)})^T \; \delta\boldsymbol{x} \; + \; \frac{1}{2}\delta\boldsymbol{x}^T \; \mathcal{H}(\boldsymbol{x}_p^{(n)}) \; \delta\boldsymbol{x} \, , \tag{B.5}$$

subject to

$$\|\delta\boldsymbol{x}\| \leq \delta_{\mathrm{p}}^{(n)} \, , \tag{B.6}$$

In Eqs. B.5 and B.6, $\delta\boldsymbol{x}$ represents the search direction, $q_p^{(n)}(\delta\boldsymbol{x})$ represents the quadratic approximation of $O(\boldsymbol{x})$ of Eq. B.1 at the nth iteration of the trust-region minimization problem, $\boldsymbol{x}_p^{(n)}$ represents the current estimate of the minimum of $O(\boldsymbol{x})$ at the nth iteration of the trust-region minimization problem, and $\delta_{\mathrm{p}}^{(n)}$ represents the radius of the trust region at the nth iteration. Also in Eq. B.5, the required values of $O(\boldsymbol{x}_p^{(n)})$, $\mathcal{G}(\boldsymbol{x}_p^{(n)})$ and $\mathcal{H}(\boldsymbol{x}_p^{(n)})$ are computed, respectively, by Eqs. B.1, B.2 and B.3 for $\boldsymbol{x} = \boldsymbol{x}_p^{(n)}$.

The hyper-sphere region defined by the constraint of Eq. B.6 represents the so-called trust region. The trust region is the region where is believed that the quadratic approximation $q_p^{(n)}(\delta\boldsymbol{x})$ properly represents the true function $O(\boldsymbol{x})$. Therefore, the minimum of $q_p^{(n)}(\delta\boldsymbol{x})$ in the hyper-sphere region represents a good estimate for the minimum of $O(\boldsymbol{x})$ in the same region.

B.1 Updating the Trust Region Radius

The trust region radius $\delta_{\mathrm{p}}^{(n)}$ is updated at each iteration based on the performance of the quadratic approximation of $O(\boldsymbol{x})$. The quality of the quadratic approximation is measured based on the mismatch between the true function $O(\boldsymbol{x})$ and the quadratic approximation, as follows [79]

$$\rho_p^{(n)}(\delta\boldsymbol{x}^*) = \frac{O(\boldsymbol{x}_p^{(n)}) \; - \; O(\boldsymbol{x}_p^{(n)} + \delta\boldsymbol{x}^*)}{q_p^{(n)}(\boldsymbol{0}) \; - \; q_p^{(n)}(\delta\boldsymbol{x}^*)} \, . \tag{B.7}$$

In Eq. B.7, $\rho_p^{(n)}(\delta\boldsymbol{x}^*)$ measures the performance of the quadratic approximation $q_p^{(n)}(\delta\boldsymbol{x})$ in Eq. B.5 at the nth iteration, and $\delta\boldsymbol{x}^*$ represents the solution of the trust region minimization problem given by Eqs. B.5 and B.6, i.e., $\delta\boldsymbol{x}^*$ is the minimizer of $q_p^{(n)}(\delta\boldsymbol{x})$ in the trust region, as we discuss later in the Section B.2. Notice that $q_p^{(n)}(\mathbf{0}) = O(\boldsymbol{x}_p^{(n)})$. As one can see from Eq. B.7, for a perfect quadratic approximation one gets $\rho_p^{(n)}(\delta\boldsymbol{x}^*) = 1.0$.

In practice, for $\rho_p^{(n)}(\delta\boldsymbol{x}^*) < \eta_1$, with the parameter $\eta_1 \in (0, 1/4]$ defined by the user, one considers that the quadratic approximation $q_p^{(n)}(\delta\boldsymbol{x})$ represents a poor approximation of the true function $O(\boldsymbol{x})$ [83]. For this case, we disregard the solution $\delta\boldsymbol{x}^*$ and repeat the trust region minimization iteration using the same estimate for the minimum of $O(\boldsymbol{x})$, i.e., $\boldsymbol{x}_p^{(n+1)} = \boldsymbol{x}_p^{(n)}$. However, we reduce the trust region radius by cutting it by half, i.e., $\delta_{\mathrm{p}}^{(n+1)} = \delta_{\mathrm{p}}^{(n)}/\,2$. Conversely, for $\rho_p^{(n)}(\delta\boldsymbol{x}^*) > \eta_1$ one considers that the quadratic approximation $q_p^{(n)}(\delta\boldsymbol{x})$ properly represents the true function $O(\boldsymbol{x})$ [83]. For this latter case, we update the current estimate of the minimum of $O(\boldsymbol{x})$ as

$$\boldsymbol{x}_p^{(n+1)} = \boldsymbol{x}_p^{(n)} + \delta\boldsymbol{x}^* \,. \tag{B.8}$$

Also, if $\rho_p^{(n)}(\delta\boldsymbol{x}^*) > \eta_2$, with the parameter $\eta_2 \in [1/2, 1]$ defined by the user, and $\|\delta\boldsymbol{x}^*\| > \delta_{\mathrm{p}}^{(n)}/\,2$, we update the trust region radius by $\delta_{\mathrm{p}}^{(n+1)} = \min(2\delta_{\mathrm{p}}^{(n)}, \delta_{\max})$. Otherwise, we repeat the same trust region radius in the next iteration, i.e., $\delta_{\mathrm{p}}^{(n+1)} = \delta_{\mathrm{p}}^{(n)}$. The parameter $\delta_{\max}$ is defined by the user and represents a limit for the maximum trust region radius. The Algorithm B.1 details the update of the trust region radius and current estimate of the minimum for the trust region minimization problem. In this dissertation, all trust region minimization problems are conducted using $\eta_1 = 1/20$ and $\eta_2 = 3/4$. In the case that the current trust region radius becomes small than a minimum value $\delta_{\min}$ defined by the user, we regard the minimization problem as converged.

B.2 Solving the Trust Region Minimization Sub-Problem

The Hessian defined by Eq. B.3 is a real symmetric positive definite matrix. Conse-

Algorithm B.1: Update the Trust Region Radius and Current Estimate of the Minimum

1. Select $\eta_1 \in (0, 1/4]$, $\eta_2 \in [1/2, 1]$ and $\delta_{\max}$.
2. Given the current estimate of the minimum, $\boldsymbol{x}_p^{(n)}$, and the current trust region radius, $\delta_{\rm p}^{(n)}$, solve the trust region minimization problem of Eqs. B.5 and B.6 to determine $\delta\boldsymbol{x}^*$ (details on Section B.2).
3. Compute $\rho_p^{(n)}(\delta\boldsymbol{x}^*)$ using Eq. B.7.
4. **If** ($\rho_p^{(n)}(\delta\boldsymbol{x}^*) < \eta_1$)
   - Set $\delta_{\rm p}^{(n+1)} = \delta_{\rm p}^{(n)} / 2$.
   - Set $\boldsymbol{x}_p^{(n+1)} = \boldsymbol{x}_p^{(n)}$

   Else If ($\rho_p^{(n)}(\delta\boldsymbol{x}^*) > \eta_2$ and $\|\delta\boldsymbol{x}^*\| > \delta_{\rm p}^{(n)} / 2$)
   - Set $\delta_{\rm p}^{(n+1)} = \min(2\delta_{\rm p}^{(n)}, \delta_{\max})$.
   - Update $\boldsymbol{x}_p^{(n)}$ using Eq. B.8.

   Else
   - Set $\delta_{\rm p}^{(n+1)} = \delta_{\rm p}^{(n)}$.
   - Update $\boldsymbol{x}_p^{(n)}$ using Eq. B.8.

quently, the quadratic approximation $q_p^{(n)}(\delta\boldsymbol{x})$ of Eq. B.5 has an unique and global minimum at [83]

$$\delta\boldsymbol{x}_{\rm global} = -\mathcal{H}(\boldsymbol{x}_p^{(n)})^{-1}\,\mathcal{G}(\boldsymbol{x}_p^{(n)})\,. \tag{B.9}$$

In the case that $\|\delta\boldsymbol{x}_{\rm global}\| \leq \delta_{\rm p}^{(n)}$, the global minimizer $\delta\boldsymbol{x}_{\rm global}$ represents the solution of the trust region sub-problem given by Eqs. B.5 and B.6, i.e., $\delta\boldsymbol{x}^* = \delta\boldsymbol{x}_{\rm global}$. Otherwise, $\|\delta\boldsymbol{x}_{\rm global}\| > \delta_{\rm p}^{(n)}$ and the solution of the trust region sub-problem has to be at the boundary of the corresponding trust region, i.e., $\|\delta\boldsymbol{x}^*\| = \delta_{\rm p}^{(n)}$. Since the Hessian is symmetric positive definite, we apply the method of Lagrange multipliers [83] to find the minimum of $q_p^{(n)}(\delta\boldsymbol{x})$ in the trust region. From Eqs. B.5 and B.6, we define the Lagrangian as [113]

$$\mathcal{L}(\delta\boldsymbol{x};\lambda) = O(\boldsymbol{x}_p^{(n)}) + \mathcal{G}(\boldsymbol{x}_p^{(n)})^T\delta\boldsymbol{x} + \frac{1}{2}\delta\boldsymbol{x}^T\,\mathcal{H}(\boldsymbol{x}_p^{(n)})\,\delta\boldsymbol{x} + \frac{1}{2}\lambda\big[\delta\boldsymbol{x}^T\delta\boldsymbol{x} - (\delta_{\rm p}^{(n)})^2\big]\,. \tag{B.10}$$

In Eq.B.10, $\lambda \geq 0$ represents the Lagrange multiplier. For the case $\lambda = 0$, the minimizer of $\mathcal{L}(\delta\boldsymbol{x};\lambda)$ in Eq.B.10 satisfy the constraint in Eq. B.6 and, similarly as before, is given by Eq.B.9. At an optimal solution we must have $\nabla\mathcal{L} = 0$ [83], which, from Eq. B.10, implies that [113]

$$\nabla_{\delta\boldsymbol{x}}\,\mathcal{L}(\delta\boldsymbol{x};\lambda) \;=\; 0 \quad \rightarrow \quad \Big(\mathcal{H}(\boldsymbol{x}_p^{(n)}) + \lambda I_{N_x}\Big)\delta\boldsymbol{x} = -\mathcal{G}(\boldsymbol{x}_p^{(n)})\,, \tag{B.11a}$$

$$\frac{\partial\mathcal{L}(\delta\boldsymbol{x};\lambda)}{\partial\lambda} \;=\; 0 \quad \rightarrow \quad \delta\boldsymbol{x}^T\delta\boldsymbol{x} = (\delta_{\rm p}^{(n)})^2\,. \tag{B.11b}$$

The Eqs. B.11a and B.11b are solved simultaneously to determine the minimizer of $\mathcal{L}(\delta\boldsymbol{x};\lambda)$ in Eq.B.10, which we shall denoted here as $\delta\boldsymbol{x}^*$ and λ^*. To solve Eqs. B.11a and B.11b, we resort to the singular value decomposition (SVD) [46] of the symmetric positive definite Hessian $\mathcal{H}(\boldsymbol{x}_p^{(n)})$ [83]

$$\mathcal{H}(\boldsymbol{x}_p^{(n)}) = U\,W\,U^T\,. \tag{B.12}$$

In Eq. B.12, the $N_x \times N_x$ matrix U denotes the orthogonal matrix of singular vectors, i.e., the columns of U represent the singular vectors $\boldsymbol{u}_j$, for $j = 1, 2, \ldots, N_x$, of the Hessian $\mathcal{H}(\boldsymbol{x}_p^{(n)})$, and the jth entry in the diagonal of the $N_x \times N_x$ diagonal matrix W contains the jth singular value, w_j, of the Hessian. Using Eq. B.12, Eq. B.11a, reduces to [83]

$$\delta\boldsymbol{x} = -\sum_{j=1}^{N_x} \frac{\boldsymbol{u}_j^T\,\mathcal{G}(\boldsymbol{x}_p^{(n)})}{w_j + \lambda}\,\boldsymbol{u}_j\,. \tag{B.13}$$

Defining $f_\lambda(\lambda) \equiv \delta\boldsymbol{x}^T\delta\boldsymbol{x}$, from Eq. B.13 one gets

$$f_\lambda(\lambda) \equiv \delta\boldsymbol{x}^T\delta\boldsymbol{x} = \sum_{j=1}^{N_x}\left[\frac{\boldsymbol{u}_j^T\,\mathcal{G}(\boldsymbol{x}_p^{(n)})}{w_j + \lambda}\right]^2\,. \tag{B.14}$$

In Eq. B.14 we use the fact that $\boldsymbol{u}_j^T\boldsymbol{u}_\ell = 0$, for any $j \neq \ell$, and that $\boldsymbol{u}_j^T\boldsymbol{u}_j = 1$.

Using Eq. B.14 into Eq. B.11b results in

$$f_\lambda(\lambda) - (\delta_{\rm p}^{(n)})^2 = 0\,. \tag{B.15}$$

We can find the solution λ^* of Eq. B.15 by defining

$$\theta(\lambda) \equiv f_\lambda(\lambda) - (\delta_{\mathrm{p}}^{(n)})^2 \,, \tag{B.16}$$

and applying the Newton-Raphson method to solve $\theta(\lambda) = 0$, i.e., to find the zero of $\theta(\lambda)$ of Eq. B.16 [93]. Using Eq. B.14, the derivative of $\theta(\lambda)$ with respect to λ is given by

$$\theta'(\lambda) \equiv \frac{\mathrm{d}\theta(\lambda)}{\mathrm{d}\lambda} = -2 \sum_{j=1}^{N_x} \frac{\left[\boldsymbol{u}_j^T \, \mathcal{G}(\boldsymbol{x}_p^{(n)}) \right]^2}{(w_j + \lambda)^3} \,. \tag{B.17}$$

For this case, $\lambda > 0$ and $w_j > 0$, for $j = 1, 2, \ldots, N_x$, consequently $\theta'(\lambda)$ in Eq. B.17 is always negative. As discussed earlier, we are considering the case that for $\lambda = 0$ the constraint in Eq. B.6 is violated. Therefore, $f_\lambda(\lambda = 0) > (\delta_{\mathrm{p}}^{(n)})^2$ (see Eq. B.14), which implies that $\theta(\lambda = 0) > 0$. Consequently, since $\theta'(\lambda) < 0$, the Eq. B.15 has an unique solution in the interval $(0, \infty)$, which is given by the unique zero of $\theta(\lambda)$ of Eq. B.16 for $\lambda > 0$. Moré and Sorensen [80] showed that redefining Eq. B.16 as

$$\theta(\lambda) \equiv \frac{1}{\delta_{\mathrm{p}}^{(n)}} - \frac{1}{\sqrt{f_\lambda(\lambda)}} \,, \tag{B.18}$$

one achieves faster convergence when applying the Newton-Raphson method. It is straightforward to verify that for $\theta(\lambda)$ in Eq. B.18, it holds that $\theta(\lambda = 0) > 0$ and $\theta'(\lambda) < 0$, thus $\theta(\lambda) = 0$ has an unique solution for $\lambda > 0$. Gould et al. [48] generalized the ideas of Moré and Sorensen [80] and proposed to redefining Eq. B.18 as

$$\theta(\lambda) \equiv [f_\lambda(\lambda)]^{\beta/2} - [\delta_{\mathrm{p}}^{(n)}]^\beta \,. \tag{B.19}$$

The main ideal of Gould et al. [48] is to choose an exponent β such that $\theta(\lambda)$ in Eq. B.19 represents an approximate linear function of λ. Consequently, one achieves faster convergence

of the Newton-Raphson method. Gao et al. [40] modified Eq. B.19 and suggested

$$\theta(\lambda) \equiv \left[\frac{\sqrt{f_\lambda(\lambda)}}{\delta_{\mathrm{p}}^{(n)}}\right]^{\beta} - 1\,. \tag{B.20}$$

Furthermore, Gao et al. [40] proposed a methodology to estimate an optimal value for the exponent β in Eq. B.20 by imposing the second derivative of $\theta(\lambda)$ in Eq. B.20 equal to zero, which is a necessary condition for $\theta(\lambda)$ behave as a linear function of λ. Nevertheless, following Rafiee [93], we adopt $\beta = -1.5$, which provides an efficient Newton-Raphson implementation for the problems considered in this dissertation. The required derivative of $\theta(\lambda)$ in Eq. B.20 is given by

$$\theta'(\lambda) \equiv \frac{\mathrm{d}\theta(\lambda)}{\mathrm{d}\lambda} = -\frac{1}{[\delta_{\mathrm{p}}^{(n)}]^{\beta}}\beta \sum_{j=1}^{N_x} \frac{\left[\boldsymbol{u}_j^T\, \mathcal{G}(\boldsymbol{x}_p^{(n)})\right]^2}{(w_j+\lambda)^3} \left(\sum_{j=1}^{N_x}\left[\frac{\boldsymbol{u}_j^T\, \mathcal{G}(\boldsymbol{x}_p^{(n)})}{w_j+\lambda}\right]^2\right)^{\beta/2-1}\,. \tag{B.21}$$

Applying the Newton-Raphson method to solve $\theta(\lambda) = 0$, for $\theta(\lambda)$ given by Eq. B.20 with $\beta = -1.5$, one finds the optimal solution λ^*. Using λ^* into Eq. B.13, one determines the optimal solution $\delta\boldsymbol{x}^*$ for the trust region sub-problem. The current estimate, $\boldsymbol{x}_p^{(n)}$, of the minimum of $O(\boldsymbol{x})$ and the current trust region radius, $\delta_{\mathrm{p}}^{(n)}$, at the nth iteration are then updated using Algorithm B.1.

The trust region algorithm is regarded as converged when no improvement of the estimated minimum of $O(\boldsymbol{x})$ is observed, i.e., when

$$\rho_{\mathrm{stop}}^{(n)} \equiv \frac{\mathrm{abs}\left[O(\boldsymbol{x}_p^{(n+1)}) - O(\boldsymbol{x}_p^{(n)})\right]}{\mathrm{abs}\left[O(\boldsymbol{x}_p^{(n+1)})\right]} \leq \epsilon_{\min}\,. \tag{B.22}$$

In Eq. B.22, "abs[·]" represents the absolute value function, $\rho_{\mathrm{stop}}^{(n)}$ represents the stop criterion, and the precision parameter $\epsilon_{\min}$ is defined by the user. Details of the trust region algorithm are presented at the Algorithm B.2.

B.3 Trust Region Algorithm for Large Scale Reservoir Problems

Algorithm B.2: Trust Region Minimization Sub-Problem

1. Select $\boldsymbol{x}_p^{(0)}$, $\delta_{\mathrm{p}}^{(0)}$, $\delta_{\min}$ and $\epsilon_{\min}$. Set $n = 0$.
2. **While** ($\rho_{\mathrm{stop}}^{(n)} > \epsilon_{\min}$ and $\delta_{\mathrm{p}}^{(n)} > \delta_{\min}$)
   - Compute $O(\boldsymbol{x}_p^{(n)})$, $\mathcal{G}(\boldsymbol{x}_p^{(n)})$ and $\mathcal{H}(\boldsymbol{x}_p^{(n)})$ using Eqs. B.1, B.2 and B.3, respectively.
   - Compute $\delta\boldsymbol{x}_{\mathrm{global}}$ using Eq. B.9.
   - **If** ($\|\delta\boldsymbol{x}_{\mathrm{global}}\| \leq \delta_{\mathrm{p}}^{(n)}$)
     - Set $\delta\boldsymbol{x}^* = \delta\boldsymbol{x}_{\mathrm{global}}$.
     - Update $\boldsymbol{x}_p^{(n)}$ and $\delta_{\mathrm{p}}^{(n)}$ using Algorithm B.1.
     - Compute $\rho_{\mathrm{stop}}^{(n)}$ using Eq. B.22.
     - Set $n = n + 1$.

     Else
     - Solve $\theta(\lambda) = 0$ (for $\theta(\lambda)$ in Eq. B.20) to find λ^*.
     - Compute $\delta\boldsymbol{x}^*$ using Eq. B.13.
     - Update $\boldsymbol{x}_p^{(n)}$ and $\delta_{\mathrm{p}}^{(n)}$ using Algorithm B.1.
     - Compute $\rho_{\mathrm{stop}}^{(n)}$ using Eq. B.22.
     - Set $n = n + 1$.

     End If

To assembly Eq. B.20 in order to solve $\theta(\lambda) = 0$, one must compute the SVD of a $N_x \times N_x$ matrix. For practical petroleum reservoir applications, the dimension N_x can be considerable large, which impacts the computational efficiency of the trust region algorithm presented above. However, the dimension N_y (e.g. see Eq. B.3) is much smaller than N_x for most practical applications. Motivated by the fact that $N_y \ll N_x$, Gao et al. [40] used the follow matrix inversion identity [89]

$$\left[C_X^{-1} + G(\boldsymbol{x})^T\, C_Y^{-1}\, G(\boldsymbol{x})\right]^{-1} = C_X - C_X\, G(\boldsymbol{x})^T \left[C_Y + G(\boldsymbol{x})\, C_X\, G(\boldsymbol{x})^T\right]^{-1} G(\boldsymbol{x})\, C_X\,, \quad \text{(B.23)}$$

to take advantage of the structure of the Hessian $\mathcal{H}(\boldsymbol{x})$ in Eq. B.3 and introduced a trust region algorithm for large scale problems. In Eq. B.23, C_X represents a given $N_x \times N_x$ symmetric positive definite real matrix, and C_Y represents a given $N_y \times N_y$ symmetric positive definite real matrix. The proof of the matrix inversion identity of Eq. B.23 can be

found in Oliver et al. [89].

Gao et al. [40] trust region algorithm is computationally efficient for the cases in which $N_y \ll N_x$. Their development starts from Eq. B.11a and uses Eq.s B.3 and B.23 to derive

$$\begin{aligned}
\delta\boldsymbol{x} &= -\Big(\mathcal{H}(\boldsymbol{x}_p^{(n)}) + \lambda I_{N_x}\Big)^{-1}\,\mathcal{G}(\boldsymbol{x}_p^{(n)}) \\
&= -\Big(I_{N_x} + G(\boldsymbol{x}_p^{(n)})^T\,G(\boldsymbol{x}_p^{(n)}) + \lambda I_{N_x}\Big)^{-1}\,\mathcal{G}(\boldsymbol{x}_p^{(n)}) \\
&= -\Big((1+\lambda) I_{N_x} + G(\boldsymbol{x}_p^{(n)})^T\,G(\boldsymbol{x}_p^{(n)})\Big)^{-1}\,\mathcal{G}(\boldsymbol{x}_p^{(n)}) \\
&= -\left[\frac{1}{1+\lambda} I_{N_x} - \frac{1}{(1+\lambda)^2}\,G(\boldsymbol{x}_p^{(n)})^T\left[I_{N_y} + \frac{1}{1+\lambda}\,G(\boldsymbol{x}_p^{(n)})\,G(\boldsymbol{x}_p^{(n)})^T\right]^{-1} G(\boldsymbol{x}_p^{(n)})\right]\,\mathcal{G}(\boldsymbol{x}_p^{(n)}) \\
&= -\frac{1}{1+\lambda}\mathcal{G}(\boldsymbol{x}_p^{(n)}) + \frac{1}{(1+\lambda)^2}\,G(\boldsymbol{x}_p^{(n)})^T\left[I_{N_y} + \frac{1}{1+\lambda}\,G(\boldsymbol{x}_p^{(n)})\,G(\boldsymbol{x}_p^{(n)})^T\right]^{-1} G(\boldsymbol{x}_p^{(n)})\,\mathcal{G}(\boldsymbol{x}_p^{(n)})\,.
\end{aligned} \tag{B.24}$$

In Eq. B.24, I_{N_y} denotes the $N_y \times N_y$ identity matrix.

Following Gao et al. [40], we rewrite Eq. B.24 as

$$\delta\boldsymbol{x} = -\frac{1}{1+\lambda}\mathcal{G}(\boldsymbol{x}_p^{(n)}) + \frac{1}{(1+\lambda)^2}\,G(\boldsymbol{x}_p^{(n)})^T\boldsymbol{z}(\lambda)\,. \tag{B.25}$$

In Eq. B.25, $\boldsymbol{s}(\boldsymbol{x}_p^{(n)})$ and $\boldsymbol{z}(\lambda)$ are defined, respectively, as

$$\boldsymbol{s}(\boldsymbol{x}_p^{(n)}) = G(\boldsymbol{x}_p^{(n)})\,\mathcal{G}(\boldsymbol{x}_p^{(n)})\,, \tag{B.26}$$

and

$$\boldsymbol{z}(\lambda) = \left[I_{N_y} + \frac{1}{1+\lambda}\,G(\boldsymbol{x}_p^{(n)})\,G(\boldsymbol{x}_p^{(n)})^T\right]^{-1}\boldsymbol{s}(\boldsymbol{x}_p^{(n)})\,. \tag{B.27}$$

From Eq. B.27, one gets

$$\boldsymbol{s}(\boldsymbol{x}_p^{(n)}) = \left[I_{N_y} + \frac{1}{1+\lambda}\, G(\boldsymbol{x}_p^{(n)})\, G(\boldsymbol{x}_p^{(n)})^T \right] \boldsymbol{z}(\lambda) \Rightarrow$$

$$\boldsymbol{s}(\boldsymbol{x}_p^{(n)}) - \boldsymbol{z}(\lambda) = \frac{1}{1+\lambda}\, G(\boldsymbol{x}_p^{(n)})\, G(\boldsymbol{x}_p^{(n)})^T \boldsymbol{z}(\lambda)\,. \tag{B.28}$$

Similarly as before, we define $f_\lambda(\lambda) \equiv \delta\boldsymbol{x}^T \delta\boldsymbol{x}$, which, using Eqs. B.25 and B.28, is given by [93]

$$f_\lambda(\lambda) \equiv \delta\boldsymbol{x}^T \delta\boldsymbol{x} = \frac{1}{(1+\lambda)^2} \mathcal{G}(\boldsymbol{x}_p^{(n)})^T \mathcal{G}(\boldsymbol{x}_p^{(n)}) - \frac{1}{(1+\lambda)^3} \Big[\boldsymbol{s}(\boldsymbol{x}_p^{(n)}) + \boldsymbol{z}(\lambda) \Big]^T \boldsymbol{z}(\lambda)\,. \tag{B.29}$$

As before, to find the optimal solution, λ^*, we solving $\theta(\lambda) = 0$, with $\theta(\lambda)$ defined by Eq. B.20 and $f_\lambda(\lambda)$ given by Eq. B.29. Note that to find $\boldsymbol{z}(\lambda)$ we solve Eq. B.27, which requires only the inverse of a $N_y \times N_y$ matrix. In order to apply the Newton-Raphson method, the required derivative of $f_\lambda(\lambda)$ with respect to λ is given by [93]

$$\begin{aligned} f'_\lambda(\lambda) \equiv \frac{\mathrm{d}f_\lambda(\lambda)}{\mathrm{d}\lambda} = & -\frac{2}{(1+\lambda)^3} \mathcal{G}(\boldsymbol{x}_p^{(n)})^T \mathcal{G}(\boldsymbol{x}_p^{(n)}) \\ & + \frac{1}{(1+\lambda)^4} \Big[\boldsymbol{s}(\boldsymbol{x}_p^{(n)})^T \left[\boldsymbol{w}(\lambda) + 2\boldsymbol{z}(\lambda) \right] + \boldsymbol{z}(\lambda)^T \left[2\boldsymbol{w}(\lambda) + \boldsymbol{z}(\lambda) \right] \Big]\,, \end{aligned} \tag{B.30}$$

where

$$\boldsymbol{w}(\lambda) = \left[I_{N_y} + \frac{1}{1+\lambda}\, G(\boldsymbol{x}_p^{(n)})\, G(\boldsymbol{x}_p^{(n)})^T \right]^{-1} \boldsymbol{z}(\lambda)\,. \tag{B.31}$$

Once λ^* is calculated, the optimal $\delta\boldsymbol{x}^*$ is computed using Eq. B.25 for $\lambda = \lambda^*$.

In this dissertation, for large scale problems, i.e., when $N_y < N_x$, we replace Eqs. B.13 and B.14, respectively, by Eqs. B.25 and B.29 and apply the trust region algorithm as described in Sections B.1 and B.2.