Optimal Reservoir Management Using Adaptive Reduced-Order Models

by

Zeid M. Alghareeb

B.S. Petroleum Engineering, The University of Tulsa, 2004
M.S. Energy Resources Engineering, Stanford University, 2009

Submitted to the Department of Civil and Environmental Engineering in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in the field of Computational Science for Energy Resources Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2015

Signature redacted

Author ....
Department of Civil and Environmental Engineering
January 29, 2015

Signature redacted

Certified by.............................
John R. Williams
Professor of Civil and Environmental Engineering
Thesis Supervisor

Signature redacted

Accepted by ....
Heidi M. Nepf
Donald and Martha Harleman Professor of Civil and Environmental Engineering
Chair, Graduate Program Committee

Optimal Reservoir Management Using Adaptive Reduced-Order Models

by

Zeid M. Alghareeb

Submitted to the Department of Civil and Environmental Engineering
on January 29, 2015, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in the field of Computational Science for Energy Resource
Engineering

Abstract

Reservoir management and decision-making is often cast as an optimization problem where we seek to maximize the field's potential recovery while minimizing associated operational costs. Two reservoir management aspects are considered here, new well placement and production controls. Reservoir simulators are at the heart of this process as they aid in identifying best field development plans. The computational cost associated with managing realistic reservoirs is however substantial due to the significant number of unknowns evaluated by the simulator as well as the number of simulations required to achieve an optimal plan—it involves hundreds to thousands of reservoir simulation runs. Reduced-order models (ROM) are considered powerful techniques to address computational challenges associated with reservoir management decision-making. In this sense, they represent perfect alternatives that trade off accuracy for speed in a controllable manner. In this work, we focus on developing model-order reduction techniques that entail the use of proper orthogonal decomposition (POD), truncated balanced realization (TBR) and discrete empirical interpolation (DEIM) to accurately reproduce the full-order model (FOM) input/output behavior. POD allows for a concise representation of the FOM in terms of relatively few variables while TBR improves the overall stability and accuracy. DEIM improves the shortcomings of POD and TBR in the case of nonlinear PDEs, i.e., saturation equation, by retaining nonlinearities in lower dimensional space. Example cases demonstrate ROMs ability to reduce the computational costs by $O(100)$ while providing close overall agreement to FOM for instances with significant difference in boundary conditions (well placements and controls).

ROMs are potentially perfect alternatives to FOMs in reservoir management intensive studies such as field development and optimization. However, ROMs presented in this thesis and the overall physics-based ROMs have the tendency to perform well within a restricted zone. This zone is generally dictated by the training simulations (with a specific set of boundary conditions) used to build the ROM. Therefore, special care is considered when implementing these training runs. To mitigate the heuristic process of implementing training runs (multiple boundary conditions training runs), we apply a trust-region approach that provides an adaptive framework to systemically retrain and update ROMs utilizing new solutions (flow) characteristics revealed during the course of the optimization run. The adaptive framework for determining the optimal well placements entails the development of a hybrid optimization algorithm, MCS-MADS, that combines positive features of both local and global optimization methods. Typical FOM is used in conjunction with MCS to globally search the optimization surface while ROMs are used in conjunction with MADS to further improve the solution quality with minimum increase in computational costs. Well production controls are optimized sequentially via gradient-based trust-region approach. ROMs in this approach replace the FOM to find optimal solutions within a trust-region (subset of the optimization space). At the end of each trust-region optimization, the accuracy of the obtained solution is assessed and the ROM is updated. Both approaches are capable of handling nonlinear constraints. They are treated using a filter-based technique.

The developed framework for adaptive ROMs is applied to two realistic field examples. The first example considers maximizing net present value (NPV) through sequentially optimizing well placements and controls while the second example considers maximizing recovery through minimizing Lorenz coefficient. Nonlinear constraints including well-to-well distance and field production limits are imposed in both examples. For all cases considered, the hybrid approach for well placement based on MCS-MADS was able to constantly provide better solution quality (up to 22% increase in NPV) when compared to standalone MCS with only 3% increase in computational costs. The incorporation of ROMs for well controls was shown to reduce computational cost by 96% with only 1% difference in solution quality when compared to FOM.

Thesis Supervisor: John R. Williams
Title: Professor of Civil and Environmental Engineering

Acknowledgments

First and foremost, I would like to express my heartfelt gratitude and appreciation to my advisor, Professor John Williams, for his full support, expert guidance, patience, and endless enthusiasm over the past four years. His wide knowledge and constructive feedback helped steer this work in the right direction. I am also inspired by his ability to extend scientific concepts to new fields. It is rare and I feel extremely lucky to learn it from him. He will always be a role model to me in terms of having great leadership skills while displaying professionalism in everything he does. It has been a privilege and a pleasure to work with such an accomplished scientist and educator, and I eagerly look forward to maintaining a strong relationship in the future.

I am grateful to my thesis committee, Professors Nafi Toksöz and Olivier de Weck, for their thoughtful and invaluable suggestions during review meetings. It is no easy task, reviewing a thesis, I am indebted for their helpful and instructive comments.

I would also like to thank my professors at MIT, especially Ruben Juanes, Gilbert Strang, Karen Willcox, and Laurent Dement for providing the most conducive and stimulating learning experience I have ever been a part of. I am very grateful to the administrative staff at the Civil and Environmental Engineering Department, especially Kiley Clapper, Kris Kipp, and Patricia Glidden, for being extremely helpful and supportive.

The community at MIT, Geonumerics laboratory, and Geospatial Data Center have provided a rich and fertile environment to study and explore new ideas. Sincere thanks to all members of the Geonumerics laboratory and Geospatial Data Center, especially Dr. Abel Sanchez, Dr. Christopher Leonardi, Dr. Bruce Jones, and Kai Pan, for the informative discussions and wonderful dinner gatherings. Special thanks to my officemates, Seonkyoo Yoon, Linsen Chong, and Joao Barbosa, for the many happy conversations and laughter we had during those drowsy afternoons and sleepless nights. I am particularly grateful to my compatriot and dear friend Abdulaziz Almuhaidib, my classmate at the University of Tulsa and now at MIT, for the almost daily lunches, stimulating futuristic discussions, and fun yet productive times.

I would like to thank Saudi Aramco and MIT Energy Initiative (MITEI) for the financial support during my PhD studies. My gratitudes go to Dr. Mohammed Al-Qahtani, Vice President of Petroleum Engineering at Saudi Aramco, and Dr. Muhammed Saggaf, President of KAPSARC, for encouraging me during my early career and supporting my decision to pursue studies at MIT.

I would not have contemplated pursing a PhD if not for my parents, Mohammed and Aljohara, who instilled within me the desire to always aim high. To my parents: thank you. I am thankful to my siblings, who made Boston their ultimate destination over the past few years.

Lastly and mostly, I am grateful to my wife Mariam and son Mohammed for taking this journey with me. Doing a PhD, especially at MIT, can be quite stressful, I am truly grateful to them for keeping me sane over the past years. Their constant love, support, and encouragement have always kept me going forward.

To my grandparents, Fahad and Abdullah, who passed away during the time of writing these lines ...

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

Global demand for energy has long been increasing. Energy consumption is expected to grow by approximately 56% in 2040, higher than it was in 2010, despite gains in energy efficiency [2]. According to the U.S. Energy Information Administration (EIA), fossil fuels including oil, natural gas, and coal remain the predominant source of energy, providing about 83% of global energy demand in 2011 (63% of global energy comes from oil and natural gas), and forecasts indicate they will continue to be significant contributors for decades to come [42]. In order to meet the rising demand, the oil and gas industry is simultaneously directing its efforts toward discovering new fields as well as increasing recovery from existing fields through best practices of reservoir management.

The oil and gas industry is mainly divided into two major sectors: upstream and downstream. The upstream sector, depicted in Figure 1.1, is commonly known as exploration and production (E&P). It includes searching for potential subsurface oil and natural gas fields, drilling of exploratory wells, and subsequently managing new field discoveries through operating wells that recover and bring crude oil and/or natural gas to the surface. The downstream sector, on the other hand, involves refining crude oil and natural gas in addition to marketing and distribution of refined products. The downstream sector provides the closest connection to everyday energy consumers through means of transportation, electricity, and other useful petroleum products.

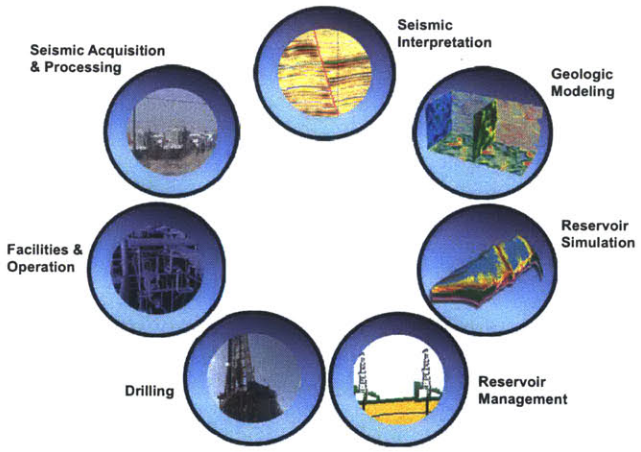


Figure 1.1: The upstream sector of the oil and gas industry. It primarily consists of eight areas covering both subsurface aspects such as drilling and reservoir management and surface aspects such as production plants and flow networks.

The main focus in this thesis is directed toward the upstream sector and specifically enhancing reservoir management and decision-making process. One of the primary goals of petroleum reservoir modeling and management processes is to enable decisions that determine the direction and course of billions of dollars every year. Such decisions are made on a long-term horizon and may include valuing acquisition of information, assigning well drilling locations, and determining development strategy for a given field. Minor improvements such as increasing recovery by 1% may result in billions of dollars—given the size of the prospective field.

A decision-making framework requires identifying a methodology to propose possible scenarios (e.g., well locations and production strategy) and developing modeling techniques to evaluate them. This can be mathematically translated into an optimization problem where optimization methods are used to propose possible scenarios whilst reservoir simulation models are used to evaluate them—simple tank models in this case are insufficient due to complex three-dimensional geometry and boundary conditions of real reservoirs. Typically, each scenario evaluation, *"sample design"*, requires performing a simulation run, and for large or complicated reservoir models, the runtime of a single evaluation can be quite substantial. In addition, the number of evaluations (hence, simulation runs) required to optimize a decision depends on the number of optimization variables—the size of the search space—and the type of optimization algorithm employed. Gradient-based optimization algorithms, for instance, might require several hundreds of simulation runs whereas the number of simulation runs for metaheuristic algorithms might be in the order of thousands. The incorporation of geological uncertainty, introduced as multiple realizations of the reservoir, further increases the computational demands and as a result hinders the popularity of optimization methods for reservoir management.

The methods presented in this thesis all aim to reduce computational complexity while preserving solution quality. There are two methods considered to increase the efficiency of the decision-making process. The first method reduces the number of scenario evaluations needed to obtain an optimal reservoir strategy; this is the aim of the field of optimization. The second method reduces the time required to obtain an evaluation for a given scenario, which is the role of reduced-order modeling (ROM).

1.2 Literature Review

The objective of this work is to develop an efficient framework to enable decision-making in reservoir management. The *Oxford English Dictionary* defines the word "efficient" as "*achieving maximum productivity with minimum wasted effort or expense*" and hence, an efficient decision-making framework should combine optimal results and computational speed. In order to achieve computational speed, we rely on both optimization techniques to reduce the number of simulation runs required to reach an optimal scenario and reduced-order modeling techniques to speedup the simulation run itself. In this section, we set up the intellectual foundation for this thesis, drawing from both academic literature and industrial practices from the field of petroleum engineering, specifically in the areas of well placement and control optimization and reduced-order modeling.

1.2.1 Well Placement Optimization

The well placement optimization problem involves varying well configurations (vertical, horizontal, or multi-lateral), well types (injector or producer) and well locations to maximize a specific objective function. The well placement problem is multi-modal—contains local optima—due to the effect of reservoir heterogeneity, and normally displays a non-smooth optimization surface. Metaheuristic methods, which avoid being trapped in local optima, are therefore mostly suitable and commonly used to solve the well placement problem, though gradient-based methods have been developed as well.

Gradient-Based Algorithms

Since analytical gradients for analytical (closed form) solutions are rarely available in reservoir simulation, numerical gradients are used to seek optimal solutions. Gradient-based algorithms used to compute gradients for the well placement problem include Simultaneous Perturbation Stochastic Approximation (SPSA) [19], Simplex Linear Interpolation (SLI) [104], and adjoint methods [89]. SPSA gradients differ from those computed using typical finite difference (FD) in the sense that all decision variables are randomly perturbed at the same time to obtain both a forward and a backward objective function evaluation and hence compute the gradient. This is superior because regardless of the number of decision variables, SPSA requires two function evaluations per iteration as opposed to FD which requires $O(n)$ function evaluations, where n is the number of decision variables [92]. At any given iteration in SPSA, a random search direction is chosen. Then, using only two objective function evaluations, forward (in the same direction) and backward, an estimate of the derivative of the objective function is obtained and hence the direction of descent is determined. A step is taken in that descent direction which is the product of the derivative and a factor that decreases with successive iterations. Bangerth et al. [19] found the SPSA method to be superior when compared to other gradient-based algorithms such as FD or gradient-free algorithms including genetic algorithms, Nelder-Mead simplex algorithm and fast simulated annealing. The SPSA algorithm, however, includes some challenges such as the choice of step size calculations to determine the next iteration.

Wang et al. [104] used a simplex linear interpolation approach to compute the objective function gradients. SLI starts by defining $n + 1$ integer vertices of a simplex that encloses the solution domain. The objective function J is computed for all vertices. SLI then generates a piecewise-linear function $\bar{J}$ from an integer-valued J and computes the gradient of $\bar{J}$ at any interior continuous point. The steepest ascent method was then used to search for optimal well placements. SLI was used within a retrospective optimization framework where a sequence of optimization subproblems that contain multiple geologic realizations were solved.

Several authors have implemented adjoint-based gradient methods for the well placement problem [112, 43, 89]. The primary advantage of using an adjoint-based method is its ability to accurately and efficiently compute gradients using only two simulation runs—a forward reservoir simulation run and a backward adjoint simulation run. Sarma and Chen [89] proposed a technique where the original discrete well location parameters (i, j) were replaced with their continuous UTMN coordinates (x, y) to obtain a continuous functional relationship between the objective function and these continuous parameters. This was achieved by replacing the point source well locations in the underlying governing PDE with a continuous representation using a bivariate Gaussian function. As a result of this transformation, the authors were able to use adjoint-based gradients to obtain optimal well locations. The efficiency and practical applicability of the approach was then demonstrated on a few synthetic waterflood optimization problems.

Zandvliet et al. [112] presented an approach where the well location to be optimized is surrounded by the so-called 'pseudowells'. These pseudowells are located in the surrounding eight grid blocks and are set to produce/inject at a very low rate and thus have a negligible influence on the overall flow. The gradients of the objective function, e.g., NPV, with respect to the pseudowell flow rates were computed using the adjoint method. These gradients were subsequently used to determine the direction of NPV improvement. The downside of this approach was the limited search direction, which was dictated by the location of the pseudowells.

Although adjoint-based gradient approaches and most gradient-based methods in general are associated with high computational efficiency, they are susceptible to getting trapped in local optima and therefore require previous knowledge of the problem domain (reservoir) to provide a good starting point.

Gradient-Free Algorithms

As stated previously, the well placement optimization problem is a high-dimensional and multi-modal with a non-smooth objective function surface. Therefore, gradient-free metaheuristic search methods such as genetic algorithms (GA), simulated annealing (SA) and particle swarm optimization (PSO) are among the most suited and commonly used methods to find optimal well placements. Once adequate number of search agents (chromosomes or swarms) is assigned, these metaheuristic methods are capable of finding global optimal solutions even if local optimal solutions exist. Although metaheuristic search methods can easily be parallelized, large problems require large search agents and therefore can be computationally prohibitive. Therefore, metaheuristic methods are usually coupled with local search methods, which have been shown to accelerate convergence toward the optimal solution with a lower number of search agents.

Bittencourt and Horne [21] optimized the placement of multiple vertical and horizontal wells using a hybrid optimization algorithm that consisted of GA, polytope and Tabu search in conjunction with a numerical reservoir simulator. Their approach was used to estimate the optimal locations of 33 wells in a reservoir with non-connected feasible regions. They indexed active cells only using a vector of active reservoir regions, arguing that (i, j) indexing is not suitable because the optimization algorithm could place wells in inactive regions. Another study investigated several types of well- block indexing for a synthetic case and discovered that (i, j) indexing of wells is more suitable for optimization algorithms since other kinds of indexing may introduce artificial noise due to discontinuities in the indexed search space [50].

Pan and Horne [81] investigated least squares and krigging interpolation algorithms to use as proxies to approximate reservoir simulation outputs for several cases, including field development optimization. They collected data for the interpolation algorithms by performing simulations on different levels of the unknowns. These levels of the unknowns were chosen by a uniform design technique. They stated that their algorithm substantially reduced the number of simulations that was required to find the global optimal solutions for the problems considered. They also found krigging to be superior to least squares for the proxy generation.

Guyaguler and Horne [50] applied a hybrid optimization algorithm that utilized GA, polytope method, krigging and artificial neural networks (used as a proxy to approximate function evaluations), along with a reservoir simulator. They optimized the locations of four vertical injection wells in a waterflood project to maximize NPV. They found krigging to be a better proxy than neural networks during the optimization.

Ozdogan and Horne [80] continued the work in [50] with the addition of time-dependent and sequential well placement process. Addressing the value of time-dependent information contributed to better decisions in term of reduced uncertainty and increased probable NPV.

A binary form of GA to optimize well type, location, and trajectory for horizontal and multi-lateral wells has been applied by Guyaguler [49] . Several helper functions were also implemented including artificial neural networks and a local hill Climber algorithm. In addition, near wellbore upscaling was applied, which approximately accounted for the effect of fine scale heterogeneity on the flow that occurred in the near-wellbore region by calculating a skin factor for each well segment. The results of this study were presented on fluvial and layered synthetic models, as well as a section model of a Saudi Arabian field.

Yeten [107] improved the preceding work and applied GA to optimize the location and type of nonconventional smart wells. He showed that using a hybridized approach including GA and a local hill climber algorithm improved the results over using GA solely. Farshi [39] converted the well placement and design optimization framework that was developed originally in [107] from a binary GA to a real-valued continuous GA. Several improvements to the optimization process were implemented such as imposing minimum distance between wells and modeling curved wellbores.

Onwunalu [79] used PSO as the underlying optimization algorithm for the well placement optimization problem. It was concluded that PSO on average provide results that are superior to those obtained using GA [51, 79]. In order to treat large-scale optimization problems involving significant number of wells, a new approach called well pattern optimization (WPO) was developed. WPO simplified the standard approach by considering repeated well patterns and then optimizing the parameters associated with the well pattern type and geometry. Finally, a metaoptimization procedure which optimized the parameters of the PSO algorithm during the optimization run was implemented. Metaoptimization involved the use of two optimization algorithms, the first algorithm optimized the PSO parameters while the second algorithm optimized the well drilling locations.

Isebor [59] devised a hybrid approach where PSO was incorporated with Mesh Adaptive Direct Search (MADS) and Branch and Bound method (B&B) to jointly determine the optimal number of wells together with their locations and controls. The results attained by the hybrid approach were shown to outperform those obtained using PSO or MADS alone in terms of the number of iterations required to achieve the optimal solution as well as the magnitude of the objective function.

1.2.2 Well Control Optimization

The well control optimization problem involves determining the optimal set of well controls, i.e., production rates and bottom hole pressure (BHP), to maximize an objective function such as NPV or recovery factor. Often, well control parameters are treated as continuous variables with the exception of smart well controls, which include discrete valve settings. A wide range of optimization techniques have been implemented for the well control optimization problem. These techniques fall into two families: gradient-based and gradient-free algorithms. Well control optimization problems are usually associated with surface and subsurface constraints. These constraints may include maximum total liquid production that can be handled by surface facilities, BHP operating range in order to not produce below the bubble point or above the fracturing pressure, and maximum gas-oil ratio. The following sections will discuss and analyze literature on gradient-based and gradient-free approaches and also the necessary techniques devised to handle constraints in details.

Gradient-Based Algorithms

The gradient-based algorithms presented in the literature to solve the well control optimization problem include steepest descent, conjugate gradient, and sequential quadratic programming (SQP) [77]. These algorithms exhibit fast convergence when compared to the gradient-free family of algorithms, especially when gradients are computed using an adjoint technique. Methods for computing gradients include FD technique which is based on the classical definition of derivates and adjoint technique which originates from optimal control theory. The advantage of using FD-based techniques is the ability to treat simulators as 'black box' since gradients are computed from function evaluations [5, 108, 10, 11]. The time required to compute the gradients in this approach is a function of the number of well controls. So, the number of simulation runs required to compute the gradients in a problem with large set of well controls might be substantial and hence computationally expensive. A partial workaround is to parallelize gradient computation as each simulation run in this case is independent. The adjoint technique on the contrary is far more efficient as it requires only two simulation runs to compute the gradient [58, 24, 88, 68, 66]. It requires one forward reservoir simulation run and one backward adjoint run regardless of the number of well controls. However, extraction of information from the reservoir simulator during the course of the computation is required, and therefore is only feasible when full access to and detailed knowledge of the simulator source code is available. As stated above, well control problems are usually associated with surface and/or subsurface constraints. These constraints may be linear, as in the case of incompressible flow where the injected water has to equal the produced fluids (oil and water); nonlinear, as in the case of maximum total liquid production; or bound constraints such as operating BHP. Techniques to handle these constraints include penalty functions, barrier methods, Lagrangian Multipliers, and filter methods [58, 59, 93].

In the context of reservoir management and well control optimization, Aitokhuehi [5] used conjugate gradient and Levenberg-Marquardt algorithm with numerical gradients to determine the control valve settings of smart wells and maximize cumulative oil recovery. The downhole sensors in this study allowed the coupling of the optimization algorithm with history matching for model update. Yeten et al. [108] described a conjugate gradient technique to maximize cumulative oil recovery from smart wells. Their optimization technique was performed over discretized time steps to ensure that earlier control settings determined for earlier time steps would not have negative effects on the objective function at later times. Alhuthali et al. [10] and Alhuthali et al. [11] applied SQP with numerical gradients to determine the optimal injection/production flow rates for each well in a water flood strategy. Their approach relied on equalizing streamline time of flight at the producing wells to maximize sweep efficiency and delay water breakthrough. A commercial simulator that has capability to internally handle constraints was used in their study.

The adjoint technique is very superior in terms of computational efficiency as it requires only two simulation runs regardless of the number of controls. Isebor [58] presented a comparative study of several optimization methods applied to solve the well control problem. The methods considered included SQP with gradients calculated using both FD and adjoint techniques, GA, and general pattern search (GPS). In the application of these methods, it was shown that derivative-free methods tend to be about an order of magnitude slower than SQP with gradients computed using an adjoint procedure. Multiple constraint handling techniques have also been presented in his study. These techniques included penalty functions, parameterless penalties, and filter approach. It was shown that the filter approach is very effective in terms of constraint handling and generality of use.

Kraaijevanger et al. [66] devised an optimal water flood design using adjoint procedures while Brouwer and Jansen [24] and Sarma [88] have demonstrated the superior capability of adjoint procedures for production optimization and closed-loop reservoir management. Li et al. [68] applied adjoint procedures for history matching.

Gradient-Free Algorithms

Gradient-free algorithms are very popular within the engineering community due to ease of implementation as they do not require access to the objective function (simulator). There are two broad categories of gradient-free algorithms: deterministic (e.g.;

polytype, simplex, and MADS [3]), and metaheuristic such as genetic algorithms, particle swarm optimization, and simulated annealing. Deterministic gradient-free methods are local optimizers, i.e., they may get trapped in local optima, and as the name implies, they don't involve randomness (they always converge to the same solution if started from the same initial guess). Metaheuristic methods, on the contrary, are global optimizers that make use of randomness to drive the optimization.

Isebor [58] implemented a comparative study to determine the ability of both deterministic and metaheuristic methods to find optimal well controls in the presence of nonlinear constraints. Deterministic methods included general pattern search (GPS) and Hooke-Jeeves direct search while metaheuristic methods included GA. The study concluded that a hybrid implementation combining GA with an efficient local search performs better than any of the individual methods. While stochastic methods in general tend to be computationally expensive, the efficiency of derivative-free methods is significantly improved through the use of surrogate-based optimization and distributed computing (parallel computing).

Alghareeb [7] focused on the reservoir engineering aspects of finding the optimal Inflow Control Valve (ICV) configurations that optimized reservoir performance parameters such as recovery factor and NPV. GA was used as the main optimization engine to find the optimal ICV configuration. GA was accompanied by a data library (proxy) to reduce the number of required simulation runs. A commercial reservoir simulator was used as the objective function evaluator that assessed how good an ICV configuration is. Several examples were presented to show the improvement in reservoir parameters made using the optimization process including real onshore and offshore field models.

Almeida et al. [12] used GA to maximize production from smart wells under operational uncertainties such as downhole valve failure.

1.2.3 Cuckoo Search (CS)

The CS algorithm is one of the recently developed algorithms in the metaheuristic family [106]. It is inspired by the breeding of cuckoo birds and combined with an efficient flight strategy exhibited by many organisms. The main characteristic of CS is its simplicity. While similar algorithms such as GA and PSO require many parameters to be adjusted which eventually dictate the performance of these algorithms, CS is controlled by only two parameters, i.e., the fraction of eggs to be abandoned at each generation and the flight step size.

To benchmark the performance of CS algorithm, Yang and Deb [106] compared CS to GA and PSO through applying all three algorithms to ten standard optimization benchmark functions. Each algorithm was repeated 100 times to minimize statistical randomness. The rate of success at finding the global minimum, and the number of function evaluations needed before the stopping criteria was met, were recorded for each function and each algorithm. For all functions considered, CS was found to outperform PSO and GA in terms of success rate and number of required objective function evaluations. They stated that the reason for this success was a good balance of local and global search.

Walton et al. [103] implemented a Modified Cuckoo Search algorithm (MCS) that provided tremendous improvements to the convergence rate. Two modifications has been made in MCS including a flight search strategy that declines as a function of generations encouraging more localized search as eggs get closer to the optimal solution, and an interaction mechanism between elite eggs to produce even better eggs carrying better solutions. A total of seven nonlinear test functions were used to benchmark the performance of MCS. The results were compared to those obtained using other metaheuristic algorithms including PSO, CS, and Differential Evolution

(DE). Each algorithm was repeated 30 times, to minimize statistical randomness, with an initial population of 20 agents. MCS was shown to perform as well as, or better than PSO, with MCS performing significantly better for some test functions. The margin by which MCS outperformed PSO was seen to increase as the number of dimensions was increased, which highlights the robustness of this method.

Yildiz [109] used CS to optimize cutting parameters in milling operations. In this operation, the goal was to maximize the total profit by adjusting the cutting speed and the feed rate of the milling process. It was found that, not only did CS find the maximum profit but also required the smallest number of functions evaluations compared to six other competing algorithms. Gandomi et al. [45] successfully used CS to optimize the structural design of 13 problems ranging from optimizing the structure of a simple beam to a full car. The performance of CS in this study outperformed GA and PSO. It was also shown that the sensitivity of the solution to CS parameters was very small. CS was also used for other applications such as breaking encrypted messages [90], training of neural networks, and developing bloom filters for database application [98].

In the field of aeronautical engineering, Walton [102] demonstrated the use of MCS to optimize the shape and design of an aerofoil. The goal was to design an aerofoil with minimum resistance (drag force) to the surrounding fluids out of a sample of 25 existing aerofoil shapes. In addition, MCS was used to optimize meshing scheme of multiple shape objects. Proper orthogonal decomposition (POD) was used along with MCS to speed up the computational time and allow to treat the optimization problem globally rather than locally.

To the best of our knowledge, both CS and MCS have never been adopted in the well placement and control optimization problems and in the field of petroleum engineering in general. Our aim here is to introduce MCS for petroleum engineering applications.

In addition, CS and MCS were used as the sole optimization methods with only unconstrained optimization problems. Thus, we will establish a hybrid approach where we combine MCS with an efficient local optimizer, namely, Mesh Adaptive Direct Search (MADS) that can handle constraints through a filter method.

1.2.4 Reduced-Order Modeling Techniques

The aim of reduced-order modeling (ROM) is to reduce the number of degrees of freedom representing a model, while retaining a sufficient level of accuracy. This is achieved by transforming high-dimensional models into lower-dimensional representations that contain dominant characteristics of the corresponding solution space to replicate the high-dimensional model's input/output behavior. The primary motivation for using ROM comes from the observation that the solution space of many numerical models is often embedded in a manifold that has much lower dimensions than the dimensions of the original, spatially descritized, models.

Reduced-order modeling has been applied to a variety of applications such as dynamic simulation, data classification, visualization, and data compression. Descriptions of the major classes of reduced-order modeling methods are presented by Antoulas and Sorensen [13], Rewieński [84], and Cardoso [28]. Our review here follows these reviews.

Most reduced-order modeling techniques project high-dimensional, referred to as *'full-order'*, states into a lower representations through the use of basis functions. There are primarily two contrasting families of techniques to construct these basis functions: equation driven techniques such as Krylov subspace and truncated-balance reduction (TBR) and empirical data driven techniques such as proper orthogonal decomposition (POD). The former techniques have been primarily used with linear time invariant models (LTI), while the latter technique has been used with nonlinear models. We briefly discuss these three techniques first within the context of linear models and then for nonlinear models.

Krylov-based methods generate reduced basis by approximating eigenvectors corresponding to the largest eigenvalues. Considering the following system of linear equations $\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu}$, the reduced basis are constructed by forming n-order Krylov subspace that is given as $\mathbf{K}_n \{\mathbf{A}, \mathbf{b}\} = span(\mathbf{b}, \mathbf{Ab}, \mathbf{A}^2\mathbf{b}, \cdots, \mathbf{A}^{n-1}\mathbf{b})$. Krylov-based algorithms are very efficient since the subspace is computed via matrix-vector multiplication rather than matrix-matrix multiplication. The columns of the Krylov subspace tend to become linearly dependent, so orthogonalization methods such as Arnoldi and Lanczos iterations are used to constitute the reduced-order basis.

Krylov projection algorithms are among the most widely used techniques for constructing reduced-order basis for LTI dynamical systems. The main reasons for their popularity is the low computational cost for large systems, which is $O(q^2N)$, and the ease of generating the projection basis. However, Krylov-based algorithms lack any provable error bounds in addition to the reduced-order basis inability to preserve the full-order system stability and passivity.

Truncated Balanced Realization (TBR) constructs the basis matrix by exploiting the structure of the system of equations [74, 48, 38, 96]. The basic idea behind TBR is to perform a change of coordinates $\hat{x} = Tx$ to a coordinate system where the states $\hat{x}$ are ordered from most important to least important and then truncate the least important states. The *"important"* states are defined as those that are either very controllable or very observable. Conversely, the least important states are those that are neither controllable nor observable. The controllability and observability of states can be quantified via the controllability and observability Grammians, respectively [26]. A system with state vector $\mathbf{x_1}$ is called *controllable* if and only if the system states can be changed by changing the control inputs. On the other hand, a particular state $\mathbf{x_1}$ is called *observable* if there exists a control input that transfers the state of the system from the initial state $\mathbf{x_0}$ to $\mathbf{x_1}$ in some finite time interval.

TBR is one of the few projection techniques that poses global error bounds on the accuracy of the resulting reduced model. However, it is limited to models with few hundred unknowns as the computational complexity is $O(N^3)$. TBR has been mainly applied in control system theory where systems are mostly static[48, 38, 96]. In the context of fluid flow, TBR has been applied to single-phase flow reservoir models by Zandvliet [111] and proposed by Heijn et al. [55] for two-phase flow reservoir models. Bui-Thanh and Willcox [26] and Rowley [85] applied TBR to fluid dynamics problems.

Proper orthogonal decomposition (POD) was first developed in 1901 as a tool to analyze coherent structures in dynamical systems [57]. POD is described as an orthogonal linear transformation that identifies an optimal coordinate system to represent an ensemble of data such that the greatest variance lies within the first coordinate, the second greatest variance lies within the second coordinates, and so on. POD yields an optimal set of coordinates such that most of the ensemble data variances are captured with only few coordinates.

POD efficiently constructs basis functions, known as spatial modes, from singular value decomposition (SVD) of *snapshots.* Snapshots are discrete samples of trajectories associated with a particular set of forcing control inputs. The constructed basis functions form a basis matrix and the size of the basis matrix is determined based on a *'relative omitted energy'* criterion where bases yielding low energy are discarded.

The reduced-order model is obtained by projecting the original full-order model onto the reduced basis, enabling a significant reduction in the number of unknowns that must be solved. The number of bases extracted using POD is optimal in the sense that, for the same number of basis functions, no other bases can represent the given snapshot set with lower least-squares error.

POD is considered one of the most popular reduced-order modeling techniques. It is easy to use and relatively requires low computational cost to generate the reduced basis from the snapshots. However, the accuracy of the generated reduced basis is dictated by how well snapshots cover the solution space. Snapshots are generated using experimental data or by running the full-order model several times using different boundary conditions. The boundary conditions that generate representative solution space may not be known a priori.

The range of applications where POD has been implemented is substantial. Bui-Thanh et al. [27] applied POD to reconstruct flow fields from incomplete aerodynamic data sets. Chaturantabut and Sorensen [33] combined POD with discrete empirical interpolation to efficiently simulate nonlinear miscible viscous fingering in porous media, and Walton [102] applied POD to a steady turbulent compressible flow passing through aerofoils. Several researchers have used POD in fluid flow through porous media problems [95, 54, 30, 93, 29, 53, 86] which will be discussed in details in Section 1.2.4.

Reduced-Order Modeling for Nonlinear Systems

There are multiple factors that can affect the efficiency of reduced-order models. Among these factors is the presence of nonlinearity. Since full-order models of interest (petroleum reservoir simulation models) are inherently nonlinear, reduced-order modeling techniques, although reduce dimensions in the sense that far fewer variables are computed, still depend on the dimension of the original full-order model through nonlinear terms [33]. This is mainly because in order to compute reduced nonlinear terms, one must first reconstruct the full-order state solution from the reduced basis, evaluate the full-order nonlinear terms, then project them onto the reduced subspace again. A similar procedure occurs when solving the reduced-order model using Newton iteration. At each iteration, beside the expense of computing the non-linear terms, the Jacobian must also be computed at the full-order dimension before projecting onto the reduced subspace and hence the overall computation cost still depends on the full-order dimension. In fact, Chaturantabut and Sorensen [33] have shown that reduced-order modeling techniques such as POD can be slower than full-order models due to such computational complexity at each iteration and each time step. To lower the computational complexity to that of the reduced subspace, several methods have been developed to recover the efficiency of reduced-order models [33, 28, 84, 44, 15]. In this section, we namely discuss three approaches, Missing Point Estimation (MPE), Trajectory Piecewise Linearization (TPWL), and Discrete Empirical Interpolation Method (DEIM).

Missing point estimation (MPE) was first introduced by Astrid et al. [15] to improve the computational efficiency of POD. The basic idea of MPE is to construct POD basis from a subset of the spatial domain rather than the entire spatial domain. In particular, a restricted POD basis is formed by extracting rows of the standard POD basis vectors corresponding to selected grid blocks. Subsequently, the subset of governing equations are projected onto the subspace spanned by these restricted POD basis. Two algorithms have been presented by Astrid et al. [15] to select the subset of grid points. Both algorithms aim to limit the growth of the condition number of the matrix formed by the selected rows of the POD basis vectors. MPE has been applied with POD to construct a reduced-order model that simulated temperature distribution in a glass melt feeder. An additional speedup factor of 5.3 has been achieved over the basic POD reduced-order model. MPE has also been applied to multiphase flow [28] and steady aerodynamics problems [99].

Trajectory piecewise linearization (TPWL) aims to reduce computational complexity of reduced-order models by approximating nonlinear terms. TPWL, developed by Rewieński [84], represents the nonlinear model as a weighted combination of piecewise linear models generated at selected points along its trajectory. It has been applied in conjunction with various reduced-order modeling techniques such as Krylov subspace [84, 46], TBR [96], and POD [28, 53]. TPWL performs very well for problems with weak nonlinearity. For highly nonlinear problems, it is difficult to accurately approximate nonlinear terms using piecewise linear representations while maintaining a relatively small number of linearized models along the trajectory. Furthermore, the selection of the training trajectories and linearization points remains an ad-hoc process [46].

Another way to address computational inefficiency associated with nonlinear models is to approximate nonlinear terms via a linear combination of their basis vectors using expansion coefficients determined using a small set of interpolation points. This is the aim of methodologies such as Empirical Interpolation Method (EIM) [20] and Best Point Interpolation Method (BPIM) [76]. In these two methods, new sets of basis vectors, different from state basis vectors, are constructed using snapshots of nonlinear terms generated during simulation of full-order models. The use of small set of interpolation points allows nonlinear terms to be evaluated only over a subset of spatial grid points rather than the whole spatial domain and hence recover the reduced-order model efficiency.

Nevertheless both EIM and BPIM accomplish a similar objective which is approximating nonlinear terms through using a linear combination of the nonlinear term basis vectors at a set of interpolation points, they both pursue different strategies for selecting the interpolation points. EIM deploys 'greedy' selection algorithm to iteratively construct a set of interpolation points in such a way that the n-th interpolation point is placed at the spatial location where the difference between the n-th basis vector and its approximation using a linear combination of the first $n-1$ basis vectors at the $n-1$ interpolation points is greatest. On the other hand, BPIM constructs the set of interpolation points by solving an n-dimensional optimization problem to minimize the least-square error between each snapshot and its approximation using n interpolation points. As a result, the obtained n points are the optimum interpolation points in the sense that they minimize the approximation error over the entire snapshot matrix. It should be noted though that the cost associated with solving the constrained optimization problem in BPIM to construct an 'optimal' set of interpolation points is significant in comparison with the sub-optimal interpolation points constructed using EIM. Researchers reported marginal improvement in accuracy of the approximation using interpolation points generated via BPIM over EIM [20, 44]. Both EIM and BPIM have been used in several applications concerning reduction of nonlinear PDEs such as simulation of wave propagation [94], inverse parameter estimation [44], and modeling of convection-diffusion reaction [64], in addition to nonaffine and parametrized PDEs [47, 75].

Chaturantabut and Sorensen [32] recently introduced Discrete Empirical Interpolation Method (DEIM), a discrete variant of EIM, that can be easily implemented on semi-discrete systems. DEIM is intended for use with discrete systems rather than its counterpart EIM which is aimed to be used with continuous systems. We thus choose to use DEIM in this work for its adaptability with discrete spatial models. DEIM has been successfully applied in various applications including neuron modeling [65], reactive flow [25], MEMS switch modeling [56], and recently in multiphase flow in porous media [93].

Reduced-Order Modeling in Reservoir Simulation

Although the development of reduced-order modeling procedures has received significant attention in recent years, relatively few studies have been conducted to investigate the application of these approaches for reservoir management and decision-making. Our intent here is to highlight and discuss in details the recent applications of ROM methods in reservoir management.

van Doren et al. [95] developed reduced-order models for nonlinear oil-water flow equations to find optimal control strategy for water flooding using adjoints. POD was used to construct the reduce-order model for the forward state equations as well as the adjoint equations. They successfully applied the ROM procedure to maximize net present value (NPV) of a heterogenous two-dimensional model containing $2,025$ grid blocks with two horizontal wells—one producer and one injector, both divided into 45 independently controlled segments. POD was used to reduce the number of unknowns from $4,050$ in the FOM to $20 - 100$ in the ROM. The speedup factor achieved for the overall optimization process was less than a factor of 1.5. The main reason for this modest runtime reduction is due to the necessity to make a full-order run at the end of the optimization to check the solution accuracy. If there is a mismatch between the optimum NPV obtained form the ROM and its counterpart obtained using the FOM, the whole optimization is repeated until convergence is achieved.

Markovinović and Jansen [72] proposed the use of reduced-order models to accelerate the Newton iterative solver for fluid flow in porous media. The governing equations were projected onto low-order space constructed from previous time step solutions using POD. Then, the governing equations were solved in the reduced space before projecting back onto the full space. The algorithm was tested on a three-dimensional model with $93,500$ grid blocks and achieved a speedup factor of about 3. It should be noted that the use of highly optimized linear solvers would lower the speedup factor as the methodology required updating the reduced-order model which required expensive SVD operation at every time step to ensure convergence.

The preceding authors achieved modest computational reduction due to presence of nonlinearities discussed in Chapter 2. To mitigate the effect of nonlinearity, Cardoso [28] implemented TPWL to approximate nonlinear terms. Nonlinear terms were linearly approximated around collection of linearization points using Taylor series. TPWL was applied to a two-phase flow model containing $24,000$ grid blocks. Reasonable accuracy and substantial runtime speedup factor of $200 - 1000$ was reported. However, highly nonlinear terms pose potential problems to TPWL since nonlinear terms are approximated within first-order accuracy. In addition, TPWL approach requires storing Jacobian matrices for both states and controls (for linearization) as well as solution states, which occupy substantial disk space as these matrices can be quite large and difficult to manipulate. Furthermore, TPWL requires output of partial Jacobian terms such as flux Jacobian and accumulation Jacobian which may not be readily available.

Suwartadi [93] used DEIM to retain nonlinearities at lower dimensional space. DEIM was combined with POD to construct ROM for two-dimensional heterogeneous two phase flow model. The ROM was used to optimize injection and production flow rates from a five-spot well pattern. Optimization runs were carried out using both ROM and FOM. Results obtained using both models in terms of NPV were in close agreement with very small relative error. The achieved runtime speedup factor was $5 - 20$.

Reduced-Order Models in Optimization

One vital aspect of reservoir management is to enable decisions that maximize petroleum recovery through optimizing well placements and controls. The well placement problem is of global nature where different locations yield different productivities. The difference in productivity and hence production rates might be of multiple orders of magnitude in the case of channelized reservoirs. The well placement problem is also of static nature where drilling locations are not updated in time. The well control problem on the other hand is a dynamic problem where surface controls or downhole controls in the case of smart wells are updated in time. The change in controls is dictated by the dynamics and behavior of fluid flowing in the porous medium, i.e., controls are manipulated to delay water breakthrough.

The use of full-order three-dimensional reservoir simulation models for such a problems is presently infeasible for large models due to high computational cost. The high computational cost hinders the popularity of optimization methods for managing petroleum reservoirs. ROMs are numerical solutions that reduce computational cost by several orders of magnitude while maintaining reasonable solution accuracy allowing optimization methods to be practically incorporated in reservoir management.

ROMs are reliable in a restricted zone around state snapshots used to construct them. This zone is commonly referred to as *"root-point"* and generally solution quality decreases as states deviate away from the root-point. Therefore, special care is required while selecting state snapshots generated from training runs as this will impact the performance of the ROM and accordingly the results of the optimization.

In the context of reservoir management, Cardoso and Durlofsky [30] applied a heuristic procedure to construct ROMs. Multiple training runs where simulated using the full-order simulation model. For each training run, input well controls such as BHPs of production wells were randomly and independently varied to cover the optimization study range and ensure that deviation from the root-point during the optimization process is minimal. The constructed ROM was used as a function evaluator in two optimization algorithms, a finite-different gradient method and gradient-free mesh adaptive search method, to maximize NPV [29]. Results for optimized NPV using ROM were shown to be very close to those achieved using the FOM, but with runtime speedup factor of 450. In addition, the constructed ROM was used for multi-objective optimization to maximize cumulative oil production while minimizing cumulative water injection [30]. Since it was not feasible to perform full-order optimization run for this multi-objective problem, only few points were simulated and the agreement between the FOM and the ROM was quite close.

The above cited researchers applied a heuristic procedure to ensure minimum deviation from the ROM root-point occurs. However, the deviation is not quite quantifiable and therefore He et al. [54] devised a more robust procedure. They applied a retraining strategy where initial ROM was used for optimization until a certain number of function evaluations was reached. The optimization run was then stopped and the ROM was retrained using the current best control variables. If a high level of discrepancy occurred between the FOM and the ROM upon retraining, the optimization run was restarted from previous control settings. The ROM update approach was incorporated in a derivative-free generalized pattern search algorithm (GPS) to optimize 36 well control settings and maximize NPV. The difference in the optimized NPV between the FOM and the ROM was shown to be 0.6% while the runtime speedups was 100.

van Doren et al. [95] applied a ROM construction and retraining strategy where initial control settings were used to construct ROM for both forward and adjoint equations.

The constructed ROM, POD-based, was then used to optimize 90 injection and production well control settings and maximize NPV. At the end of the optimization run, a full-order simulation run was performed using the optimal control settings to verify the solution accuracy. Then, if required, a new ROM was constructed, and the process was repeated until convergence of NPV obtained from ROM and FOM was achieved. The runtime speedup factor achieved using ROM was less than 1.5 due to the frequent repetition of ROM construction and optimization run when convergence was not achieved.

All of the above mentioned approaches are suboptimal in the sense that there is no indication of how many training runs are required initially to cover the control variable space nor error quantification is available to determine suitably when to retrain the ROM during the optimization run. Trust-region methods offer an effective way to manage ROM retraining over the course of the optimization run without the overburden of continuing the whole optimization run prior to error assessment, [37, 35]. The basic idea of trust-region approach for constructing ROM is to perform the optimization run over a restricted control variable space in which ROM is supposedly accurate with minimum deviation from root-point. Thus, the trust-region method ensures that the optimization algorithm always stays close to the root-point. The ROM is updated at the end of each trust-region optimization step [14, 6, 4, 93].

Agarwal and Biegler [4] implemented trust-region strategy for ROM construction and retraining. ROM was constructed using POD to optimize a two-bed isothermal pressure swing adsorption system for CO_2 capture. They implemented penalty-based trust-region and filter-based trust-region algorithms for constraint handling. Accurate results were achieved using the trust-region algorithm for ROM retraining with different computational time associated with each constraint handling method. Unfortunately, comparison of computational time between FOM and ROM was not reported.

Suwartadi [93] used a trust-region approach for ROM retraining. ROM was constructed using POD and DEIM for forward equations and vanilla POD for adjoint equations to optimize injection and production flow rates from a five-spot well pattern. The trust-region algorithm was used with Lagrangian barrier method to handle constraints. Accurate results were achieved using ROM while abiding by the constraints.

1.3 Scope of Work

The development of reduced-order modeling procedures in the field of reservoir engineering has recently received significant attention as a viable alternative to full-order simulation models. However, relatively few studies have focused on the coupling of ROM construction techniques and optimization algorithms for the well placement and control problems. The coupling entails the incorporation of ROM construction and updating techniques into the optimization algorithms. Most studies have focused on developing techniques to construct ROMs. However, training was achieved via an ad-hoc procedure where ROM training was accomplished separately from optimization. This kind of approach lacks the ability to utilize new information gained during optimization iterations to improve the accuracy of the ROM and hence the resulting optimal solutions may significantly differ from those obtained using the FOM.

This thesis is devoted to developing and applying new procedures for incorporating ROM in the optimization process for generalized field development problems. Though this work is primarily directed toward reservoir modeling and simulation, the techniques devised are general and can easily be used for other engineering applications.

To accomplish the general aim of this study, we first develop techniques to construct ROMs for fluid flow in porous media. The techniques entail the use of proper orthogonal decomposition (POD), truncated balanced realization (TBR), and discrete empirical interpolation (DEIM) to accurately reproduce the input/output behavior of the FOM. We then devise a hybrid optimization algorithm that is capable of both local and global search. A local optimization technique— namely Mesh Adaptive Direct Search (MADS), is coupled with Modified Cuckoo Search (MCS)—a global metaheuristic technique. This hybrid optimization scheme is capable of handling nonlinear constraints using a filter approach, as described in [40]. The filter approach, to the best of our knowledge, has not been previously incorporated into MCS. Finally, we introduce techniques that enable ROM construction, with updates guided by information obtained during the course of the optimization. ROM updates are based on a trust-region strategy for the well control problem and a MADS polling strategy for the well placement problem.

The key research objectives of this dissertation are:

- To further the development of reduced-order modeling techniques for multiphase flow in porous media. Specifically, we introduce a two-step reduction procedure that entails the use of POD and TBR to construct a more stable orthogonal basis matrix.
- To develop and apply DEIM procedures for multiphase flow in porous media. Specifically, we expand the capability of DEIM method in [31] to handle three-dimensional reservoir simulation models with gravitational effects.
- To introduce and enhance MCS for petroleum engineering applications. MCS is used to solve the well placement problem. A filter-based MCS is developed in this work to solve the well placement problem in the presence of nonlinear

constraints.

- To efficiently apply the developed ROM techniques to the sequentially coupled well placement and control optimization problem. We devise a ROM construction and update framework based on information obtained during the course of the optimization. ROM training and update is based on a trust-region strategy for the well control problem and a MADS polling strategy for the well placement problem.

- To accentuate the benefit of using the novel ROM-based optimization framework for well placement and control optimization on two realistic reservoir simulation models. The first model represents a cross-section from a giant field in the Middle East, while the second model represents the top five layers of SPE 10th comparative study, [34].

1.4 Dissertation Outline

This dissertation focuses on enabling the use of ROM for generalized petroleum field optimization. Figure 1.2 depicts a graphical thesis road map. In Chapter 2, we describe the underlying reduced-order modeling approach. The approach entails the use of POD, TBR, and DEIM to construct a more concise reservoir simulation model that is far less computationally demanding and still able to represent the general fluid flow behavior. The reduced-order model is capable of handling three-dimensional flow under gravitational effects. The approach is implemented in the MATLAB Reservoir Simulation Toolbox (MRST) [69]. The developed ROM procedure is applied to a heterogeneous model containing 13,200 grid blocks and five wells (one injector and four producers). The accuracy of the ROM is demonstrated for several testing simulations in which the injection and production rates for each well differ from those used to build the ROM.

The aim of Chapter 3 is to build a framework that enables the use of ROMs for well placement and controls. We begin by reviewing the main optimization algorithms for both the well placements and control problems. The optimization algorithms considered for the well placement problem include Modified Cuckoo Search (MCS), which is an improvement over the original Cuckoo Search, Mesh Adaptive Direct Search (MADS), and the MCS-MADS hybrid method, which is a variant of the PSO-MADS algorithm introduced by Isebor et al. [61]. The well control problem is solved using a simple gradient approach, with gradients computed using adjoint procedures. We then describe a filter approach to handle nonlinear constraints. The filter approach is implemented for each of these different methods. Finally, we present a framework that enables a systematic use of ROM for the well placement and control problems. The framework consists of an optimization module to find optimal solutions and a ROM construction and assessments module that builds ROMs and ensure their solution quality.

The purpose of Chapter 4 is to demonstrate the application of the developed framework on two realistic field models. The models contain 25,194 and 66,000 grid blocks, respectively, and constitute two-phase flow in a heterogeneous medium.

In Chapter 5, we conclude with a discussion on future research directions for the reduced-order modeling technique developed in this work, in addition to possible extension of the application of reduced-order modeling for general field development.

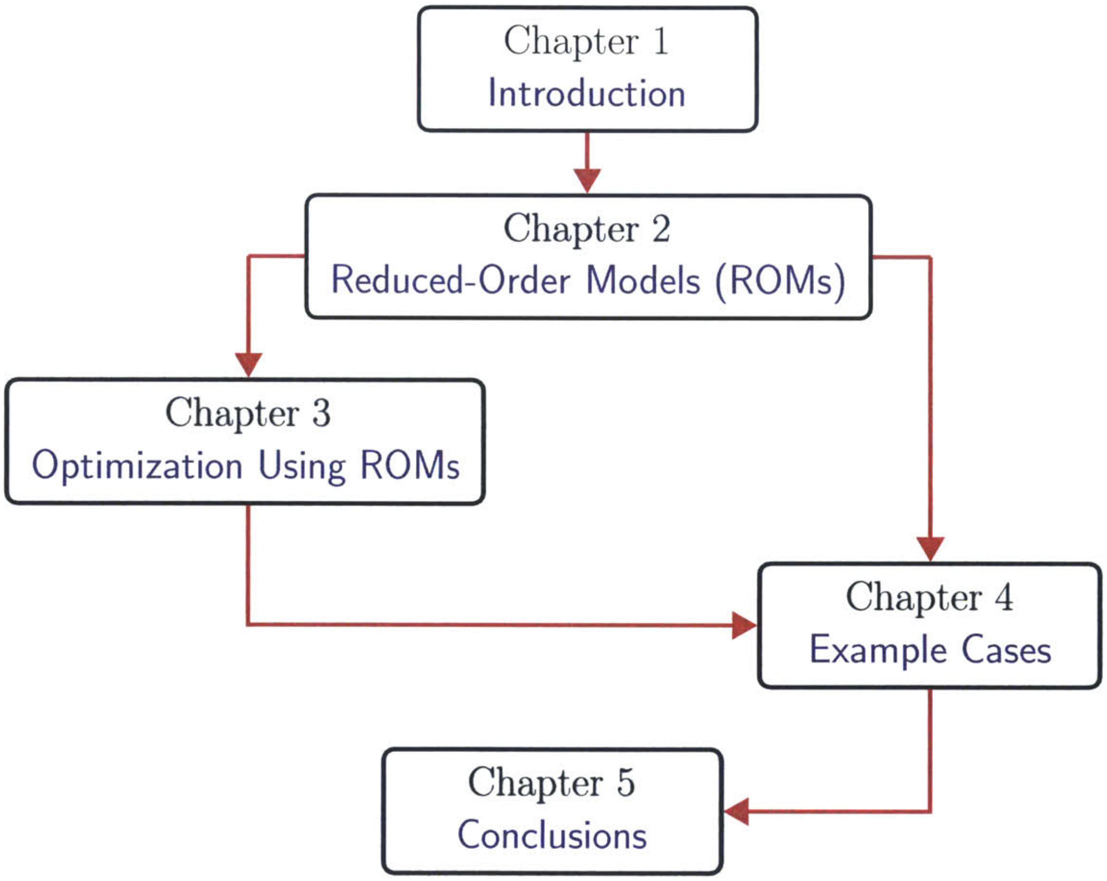


Figure 1.2: A graphical dissertation road map highlighting chapter contributions and connections.

Chapter 2

Reduced-Order Modeling of Multiphase Flow in Porous Media

This chapter discusses the development of reduced-order models for multiphase flow in porous media, i.e., reservoir simulation. It begins by describing the constitutive mass balance equations for two-phase incompressible flow, then outlines the discretization procedure and solution strategy. Section 2.2 discusses the development of reduced-order modeling techniques that include POD, TBR, and DEIM in reservoir simulation for three-dimensional models with gravitational forces. The approach combines lower computational cost with enhanced numerical stability while maintaining nonlinearities at a reduced dimension. The developed reduced-order modeling technique is applied to a heterogeneous 13,600 grid block model in Section 2.3. Results demonstrate the robustness and computational speedup of ROM techniques for realistic cases. Detailed error analysis and techniques to improve the accuracy of DEIM is presented in Section B.5 of Appendix B.

2.1 Reservoir Modeling Procedure

In this section, we describe the governing equations—including the formulation of the problem, the discretization procedure, the solution strategy, and the reduced-order modeling techniques for three-dimensional two-phase flow reservoir model with gravitational forces.

2.1.1 Governing Equations for Two-phase Flow

Reservoir simulation models are derived by combining the constitutive mass balance equation with multiphase Darcy's law. We consider incompressible oil-water flow system and neglect capillary pressure effects (production always remains above bubble point). Also, there is no mass transfer between phases, i.e., the oil component resides only in the oil phase while the water component resides only in the water phase. Then, the continuity equation for each phase, designated j (where $j = o$ for oil and w for water), is given by

$$\frac{\partial}{\partial t}(\phi \rho_j S_j) + \nabla \cdot (\rho_j \mathbf{v}_j) = q_j, \tag{2.1}$$

where ϕ is porosity, ρ_j and S_j are the density and saturation of phase j respectively, $\mathbf{v}_j$ is the phase Darcy velocity, and q_j is the source term. Assuming incompressible flow, i.e., ϕ and ρ_j are constants, The continuity equation (2.1) is simplified into:

$$\phi \frac{\partial S_j}{\partial t} + \nabla \cdot (\mathbf{v}_j) = \frac{q_j}{\rho_j}. \tag{2.2}$$

A more tractable system of equations consisting of a pressure equation and a saturation (fluid-transport) equation can be written for easier dimensionality reduction.

The pressure equation (assuming no capillary effects) is given by

$$\nabla \cdot \mathbf{v}_t = q_t, \qquad \mathbf{v}_t = -\mathbf{K}\Big[\lambda_t \nabla p + (\lambda_w \rho_w + \lambda_o \rho_o) g \nabla z\Big], \tag{2.3}$$

where $\mathbf{v}_t$ is the total Darcy velocity, $\mathbf{K}$ is the absolute permeability (assumed to be a diagonal tensor), $\lambda_t = \lambda_w + \lambda_o$ is the total mobility, $\lambda_j = k_{r_j}/\mu_j$ is the mobility of phase j, μ_j is the respective phase viscosity, k_{r_j} is the relative permeability of phase j, p is pressure, $q_t = q_w + q_o$ is the total flow rate, g is the gravitational constant, and ∇z is the negative upward vertical direction. The derived saturation equation is given by

$$\phi \frac{\partial S_w}{\partial t} + \nabla \cdot \Big(f_w(S_w) \big[\mathbf{v}_t - \lambda_o(\rho_w - \rho_o) g \mathbf{K} \nabla z\big] \Big) = q_w, \tag{2.4}$$

where $f_w(S_w) = \lambda_w/(\lambda_w + \lambda_o)$ denotes the fractional flow of water. The term $f_w(S_w)\mathbf{v}_t$ represents viscous forces while the term $f_w(S_w)\lambda_o(\rho_w - \rho_o)g\mathbf{K}\nabla z$ represents gravitational forces.

2.1.2 Discretization and Solution Strategy

The two-phase flow description for incompressible flow and no capillary effects entails three equations and three unknowns (p, S_o, S_w). We select p and S_w as the primary unknowns. Once these primary unknowns are computed, S_o can be readily determined from the saturation constraint equation, i.e., $S_w + S_o = 1$. For the sake of brevity, we let $S_w = S$ as the water saturation.

The pressure equation (2.3) and saturation equation (2.4) are nonlinearly coupled through the saturation-dependent mobilities in the pressure equation and through the pressure-dependent total velocity in the saturation equation, in addition to other terms that depend on pressure, e.g, viscosities. Following procedures developed by Lie et al. [69], a sequential method is applied to obtain solution states where saturation from previous step (or initial condition) is used to compute the saturation-dependent coefficients, e.g., λ_t in (2.3), before it is solved for pressure and subsequently total velocity. Then, total velocity is kept constant while saturation from (2.4) is solved and advanced in time. Next, the new saturation states are used to update the saturation-dependent terms in (2.3) and pressure states are solved again, and so on. The pressure equation in (2.3) is discretized explicitly while the saturation equation in (2.4) is discretized implicitly in time as follows:

$$\mathbf{T}^n \mathbf{p}^{n+1} - \boldsymbol{G}^n = \mathbf{B}\mathbf{u}^{n+1}. \tag{2.5}$$

Here $\mathbf{T}^n \equiv \mathbf{T}(S^n)$ is a diagonal transmissibility matrix that relates flow in phase j to difference in pressure. It is calculated using a two-point flux approximation scheme (see [18] for details). The superscript n represents time step, $\boldsymbol{G}^n \equiv \boldsymbol{G}(S^n)$ is a vector containing the gravitational effects, $\mathbf{u}$ is the input controls (boundary conditions), i.e., well flow rates or bottom hole pressure (BHP), $\mathbf{B}$ is the arrangement matrix for the controls and $\mathbf{p}$ is the unknown pressure vector we seek to solve. The discretized pressure equation (2.5) is a linear equation as the transmissibility matrix $\mathbf{T}^n$ does not depend on pressure (since viscosity is constant for incompressible flow). The saturation equation is discretized using finite volume method:

$$\mathbf{S}^{n+1} = \mathbf{S}^n + \frac{\Delta t}{\phi}\Big[\mathbf{F}^{n+1} + \mathbf{Q}_w\Big]. \tag{2.6}$$

Here $\mathbf{Q}_w$ denotes the source/sink vector and $\mathbf{F}^{n+1} \equiv \mathbf{F}(S^{n+1})$ is the numerical approximation of the total flux from viscous and gravitational forces across all grid block interfaces, e.g., for a grid block Ω_i and associated interfaces γ_{ij} (where ij represents the shared interface between the grid block Ω_i and the neighboring grid blocks Ω_j):

$$F_i^{n+1} = \int_{\Omega_i} \int_{\gamma_{ij}} f_w(S^{n+1})_{ij} \left[v_{ij} - g_{ij}(S^{n+1}) \right] \mathrm{d}V. \tag{2.7}$$

$f_w(S^{n+1})_{ij}$ designates the fractional flow function associated with γ_{ij}, v_{ij} is the Darcy flux, and $g_{ij}(S^{n+1})$ is the gravitational flux across the interface. The total flux term (referred to as flux in this work) can be seen to introduce nonlinearity in (2.6) as it is a function of the saturation state we seek to solve. It is also clear from (2.7) that the flux term introduces a direction dependency into the system. It is therefore treated using *"upstream weighting"*. Specifically, the evaluation of the flux term depends on the direction of flow, as follows:

$$f_w(S^{n+1})_{ij} = \begin{cases} f_w(S_i^{n+1}) & \text{if } v_{ij} \geq 0, \\ f_w(S_j^{n+1}) & \text{if } v_{ij} < 0. \end{cases} \tag{2.8}$$

Therefore, the nonlinear flux vector in (2.6) represents a non-componentwise function as the computation of each element depends on other spatially neighboring elements due to upstream weighting. If the flux term is evaluated without being upstream weighted, the numerical solution may display oscillations, overshoots, or undershoots (e.g., saturation less than zero or greater than one), or converge to an incorrect solution.

The source/sink term, right hand side in (2.4), represents wells which are the typical boundary condition in reservoir simulation. Wells are modeled using the following well equation:

$$-(q_t)_i^{n+1} = (\lambda_t)_i^n WI(p_i^{n+1} - p_i^w), \tag{2.9}$$

where $(q_t)_i^{n+1}$ is the total volumetric flow rate from block i into the well (or vise versa) at time $n+1$, p_i^{n+1} is the grid block pressure and p_i^w is the wellbore pressure for well w in grid block i, and WI is the well index. For a vertical well that fully penetrates block i, WI is computed using Peaceman well model [82]:

$$WI = \left[\frac{2\pi k \Delta z}{\ln(r_0/r_w)}\right]_i, \tag{2.10}$$

where r_w is the wellbore radius and $r_0 \approx 0.2\Delta x$. Note that if BHP is specified as a well control, it is represented in the simulator by specifying p_i^w in the well equation (2.9).

The discretized pressure equation in (2.5) represents a linear system as all parameters are independent of pressure—transmissibility is a function of saturation in the case of incompressible flow. This system can be solved using any efficient linear routine, i.e., MATLAB's built-in solver [73] for small models or AGMG [78] for large models. In contrast, the discretized saturation equation (2.6) represents a nonlinear set of algebraic equations as the flux term is a function of saturation. Thus, it is solved using Newton Raphson's method. The residual form of (2.6) is expressed as:

$$\mathbf{g} = \mathbf{S}^{n+1} - \mathbf{S}^n - \frac{\Delta t}{\phi}\left[\mathbf{F}^{n+1} + \mathbf{Q}_w\right] = 0, \tag{2.11}$$

and the Jacobian matrix is represented as:

$$\mathbf{J} = \mathbf{I} - \frac{\Delta t}{\phi}\left(\frac{\partial \mathbf{F}^{n+1}}{\partial \mathbf{S}^{n+1}}\right). \tag{2.12}$$

2.2 Reduced-Order Modeling Representation

This section describes how to reduce the pressure equation (2.5) using an approach that merges POD and TBR to formulate stable reduced pressure basis. DEIM is then applied to the nonlinear saturation equation (2.6) providing cost-efficient representation of nonlinearities. Explanation of the shortcomings when applying POD solely to nonlinear equations and how this is mitigated using DEIM is also discussed.

2.2.1 Reduced-Order Pressure Equation

Projection-based techniques are commonly used for constructing reduced-order system of equations. They construct reduced-order systems of order $\ell \ll n$ that approximates the original systems from a subspace spanned by a *reduced basis* of dimension ℓ in $\mathbb{R}^n$. Galerkin projection is used here as the means for dimension reduction. It enables the representation of the pressure state vector $\mathbf{P}$ in terms of a reduced pressure coefficient vector $\tilde{\mathbf{P}}$ using a right basis matrix $\mathbf{\Phi}_R$; i.e.,

$$\mathbf{P} = \mathbf{\Phi}_R \tilde{\mathbf{P}}. \tag{2.13}$$

The basis matrix $\mathbf{\Phi}_R$ is the key to the accuracy and stability of the ROM and can be constructed either using POD or TBR with different computational complexity associated with each approach. POD constructs the basis matrix from singular value decomposition (SVD) of *snapshots*, which are defined as discrete samples of trajectories associated with a particular set of inputs. Each snapshot corresponds to a vector of pressure values for each grid block at a given time step generated using a certain flow rate or BHP control. We denote $\mathbb{P} \in \mathbb{R}^{n \times n_s}$ as the pressure snapshot matrix where n is the number of grid blocks and n_s is the number of snapshots such that:

$$\mathbb{P} = \begin{bmatrix} \mathbf{P}^1, & \mathbf{P}^2, & \cdots & \mathbf{P}^{n_s} \end{bmatrix}. \tag{2.14}$$

The implemented POD procedure is very well established and discussed in a variety of applications (e.g., [33, 28]). The development here follows the work of Suwartadi [93]. The snapshot matrix, obtained from the FOM, is standardized by subtracting the mean from each snapshot:

$$\hat{P} = \frac{1}{n_s}\sum_{i=1}^{n_s} \mathbf{P}^i, \tag{2.15}$$

$$\hat{\mathbb{P}} = \begin{bmatrix} \mathbf{P}^1 - \hat{P}, & \mathbf{P}^2 - \hat{P}, & \cdots & \mathbf{P}^{n_s} - \hat{P} \end{bmatrix}. \tag{2.16}$$

POD formulates the generation of basis vectors as a minimization problem. Let $\mathbf{\Phi}_R = \mathbf{\Phi}$ for the sake of brevity, the basis matrix $\mathbf{\Phi} = \{\phi_i\}_{i=1}^q$ is derived such that the sum of the least-square approximation errors of the n_s snapshots is minimized:

$$\begin{aligned} \mathbf{\Phi} = \arg \min_{\{\varphi_i\}_i^q} \sum_{j=1}^{ns} \|\hat{\mathbb{P}}_j - \sum_{i=1}^{q} (\hat{\mathbb{P}}_j^T \varphi_i)\varphi_i\|_2^2, \\ \text{subject to } \varphi_i^T \varphi_j = \delta_{ij} \quad \text{for } 1 \leq i, j \leq q, \end{aligned} \tag{2.17}$$

where δ_{ij} is the Kronecker delta and we assume that $n > n_s$, i.e., the state dimension is always larger than the number of snapshots. The solution to (2.17) is given by the left singular vector of the snapshot matrix $\hat{\mathbb{P}}$. Performing SVD to $\hat{\mathbb{P}}$, the left singular vectors provide the columns of the basis matrix $\mathbf{U}$ such that:

$$\hat{\mathbb{P}} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T, \tag{2.18}$$

where $\mathbf{U} = \begin{bmatrix}\mathbf{u}_1, \cdots, \mathbf{u}_{n_s}\end{bmatrix}$ and $\mathbf{\Sigma} = \text{diag}(\sigma_1, \cdots, \sigma_{n_s})$, where $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_q \geq \cdots \geq \sigma_{n_s} \geq 0$. The POD basis are the first q dominant left singular vectors, i.e., $\mathbf{\Phi} = \{\phi_i\}_{i=1}^{q} = \{\mathbf{u}_i\}_{i=1}^{q}$. The sum of the least-square errors when approximating the n_s snapshots using the q POD basis is given by

$$\epsilon_{\text{POD}} = \sum_{j=1}^{ns} \|\hat{\mathbb{P}}_j - \sum_{i=1}^{q} (\hat{\mathbb{P}}_j^T \varphi_i)\varphi_i\|_2^2 = \sum_{i=q+1}^{n_s} \sigma_i^2. \tag{2.19}$$

This error represents the *'omitted energy'* of the snapshots due to truncation. The total energy of the snapshots is $\sum_{i=1}^{n_s} \sigma_i^2$. This energy criterion can be used to selectively retain left singular vectors, meaning that those left singular vector corresponding to small amounts of energy can be safely discarded. In particular, the number of retained basis, q, is chosen such that the *'relative omitted energy'* is less than a certain threshold:

$$\Theta = 1 - \frac{\sum_{i=1}^{q} \sigma_i^2}{\sum_{i=1}^{n_s} \sigma_i^2} < \epsilon. \tag{2.20}$$

Although POD has gained prominence over other methods such as TBR due to its lower computational complexity, it does not provide a guaranteed stability bounds. In this context, stability means that an error in approximating pressure states at time n, $\mathbf{P}^n$, does not get amplified at subsequent time steps, i.e., $\mathbf{P}^{n+1}$. Several investigators have studied stability of POD based models (see e.g., [22, 52]).

TBR constructs the basis matrix by exploiting the structure of the system of equations. It analyzes the response of the states of a system for a given control input and measures its behavior through the so-called controllability and observability concepts. To illustrate further, let's consider the following linear state-space system of equations:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \\ \mathbf{y}(t) &= \mathbf{C}\mathbf{x}(t), \end{aligned} \tag{2.21}$$

where $\mathbf{u}(t) \in \mathbb{R}^m$ is a vector containing m external forcing inputs, $\mathbf{y}(t) \in \mathbb{R}^p$ is the vector of outputs, and $\mathbf{x}(t) \in \mathbb{R}^n$ is the state vector. The matrices $\mathbf{A} \in \mathbb{R}^{nxn}$, $\mathbf{B} \in \mathbb{R}^{n\times m}$ and $\mathbf{C} \in \mathbb{R}^{p\times n}$ have constant coefficients evaluated at steady-state conditions. The second equation in (2.21) is referred to as the output equation (assuming no feedthrough terms). In the context of reservoir engineering, it is only possible to measure pressure states and flow rates at well locations and hence the matrix $\mathbf{C}$ contains coefficients at the locations where wells intersect grid blocks. Applying an input control $u = \delta$ to (2.21), the output response is denoted by $h(t) = \mathbf{C}e^{\mathbf{A}t}\mathbf{B}, t \geq 0$. This response can be decomposed into an input-to-state map $x(t) = e^{\mathbf{A}t}\mathbf{B}$, and a state-to-ouptut map $\eta(t) = \mathbf{C}e^{\mathbf{A}t}$. Thus, the input δ causes the state $x(t)$, while the initial condition $x(0)$ causes the output $y(t) = \eta(t)x(0)$. The Grammians corresponding to x and η are:

$$\mathbf{G}_c = \sum_t x(t)x(t)^T = \int_0^\infty e^{\mathbf{A}t}\mathbf{B}\mathbf{B}^T e^{\mathbf{A}^T t}\, \mathrm{d}t, \tag{2.22}$$

$$\mathbf{G}_o = \sum_t \eta(t)^T \eta(t) = \int_0^\infty e^{\mathbf{A}^T t}\mathbf{C}\mathbf{C}^T e^{\mathbf{A}t}\, \mathrm{d}t, \tag{2.23}$$

where $\mathbf{G}_c$ and $\mathbf{G}_o$ are called the controllability and observability Grammians respectively. A system with state vector $\mathbf{x}_1$ is called *controllable* if and only if the system states can be changed by changing the control inputs. On the other hand, a particular state $\mathbf{x}_1$ is called *observable* if there exists a control input that transfers the state of the system from the initial state $\mathbf{x}_0$ to $\mathbf{x}_1$ in some finite time interval. The largest singular values of the product of the controllability and observability Grammians, also known as *Hankel singular values*, describe the most controllable and observable states in the system. To compute Hankel singular values, we first determine the Grammians through solving the following Lyapunov equations:

$$\mathbf{A}\mathbf{G}_c + \mathbf{G}_c\mathbf{A}^T + \mathbf{B}\mathbf{B}^T = 0, \tag{2.24}$$

$$\mathbf{A}^T\mathbf{G}_o + \mathbf{G}_o\mathbf{A} + \mathbf{C}^T\mathbf{C} = 0. \tag{2.25}$$

Then, the Hankel singular values of the system are the square root of the eigenvalues of the product $\mathbf{G}_c\mathbf{G}_o$, i.e.,

$$\sigma = \sqrt{\lambda(\mathbf{G}_o\mathbf{G}_c)}. \tag{2.26}$$

The linear system in the state-space form is called *'balanced'* if the solutions of the two Grammians are both equal to the diagonal matrix of the Hankel singular values:

$$\mathbf{G}_c = \mathbf{G}_o = \Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \cdots, \sigma_n), \tag{2.27}$$

where $\sigma_1 \geq \cdots \geq \sigma_n \geq 0$. Typically, the Hankel singular values of the system contain useful information about the input-output behavior of the system. In particular, small Hankel values correspond to internal sub-systems that have weak effect on the input-output behavior and are almost non-observable or non-controllable or both.

Every controllable and observable system can be transformed into a balanced system by applying a basis change $\tilde{x} = \mathbf{M}x$. The two Grammians are transformed by congruence transformations:

$$\mathbf{G}_c = \mathbf{M}\mathbf{G}_c\mathbf{M}^T, \tag{2.28}$$

$$\mathbf{G}_o = \mathbf{M}^{-T}\mathbf{G}_o\mathbf{M}^{-1}, \tag{2.29}$$

$$\mathbf{G}_c\mathbf{G}_o = \mathbf{M}\mathbf{G}_c\mathbf{G}_o\mathbf{M}^{-1}. \tag{2.30}$$

The key here is to find the transformation matrix $\mathbf{M}$ such that the transformed Grammians are balanced, i.e., a transformation that makes $\mathbf{G}_c$ and $\mathbf{G}_o$ both equal to the Hankel singular values. To obtain $\mathbf{M}$, we first perform Cholesky factorization on the Grammians as follows:

$$\mathbf{G}_o = \mathbf{L}_o\mathbf{L}_o^T, \tag{2.31}$$

$$\mathbf{G}_c = \mathbf{L}_c\mathbf{L}_c^T, \tag{2.32}$$

where $\mathbf{L}_o$ and $\mathbf{L}_c$ are lower triangular matrices. Then, perfuming singular value decomposition such that $\mathbf{L}_o^T\mathbf{L}_c = \mathbf{U}\Sigma\mathbf{V}$, the balancing transformation matrix can be obtained as:

$$\mathbf{M} = \mathbf{L}_c\mathbf{V}\Sigma^{-1/2}, \tag{2.33}$$

$$\mathbf{M}^{-1} = \Sigma^{-1/2}\mathbf{U}^T\mathbf{L}_o^T, \tag{2.34}$$

Finally, under similarity transformation of the state-space model, the reduced matrices are:

$$\tilde{\mathbf{A}} = \mathbf{M}^{-1}\mathbf{A}\mathbf{M}, \tag{2.35}$$

$$\tilde{\mathbf{B}} = \mathbf{M}^{-1}\mathbf{B}, \tag{2.36}$$

$$\tilde{\mathbf{C}} = \mathbf{C}\mathbf{M}, \tag{2.37}$$

where $\tilde{\mathbf{A}}, \tilde{\mathbf{B}}, \tilde{\mathbf{C}}$ are truncated according to the least Hankel singular values.

To reduce the linear pressure equation using TBR, we apply Algorithm 1 which summarizes the TBR approach explained previously to obtain a left transformation matrix $\mathbf{W}_L$ and a right transformation matrix $\mathbf{W}_R$. $\mathbf{T}$ and $\mathbf{B}$ are the transmissibility and input control arrangement matrices in (2.4) respectively. The matrix $\mathbf{C}$ is the observation matrix with ones at the well locations and zero elsewhere.

Algorithm 1: TBR algorithm

```
Input  : System matrices $\mathbf{T}, \mathbf{B}, \mathbf{C}$
Output: Projection bases $\mathbf{W}_L$ and $\mathbf{W}_R$
1 Find controllability Grammian $\mathbf{G}_c$:
      $\mathbf{T}\mathbf{G}_c + \mathbf{G}_c\mathbf{T}^T = -\mathbf{B}\mathbf{B}^T$;
2 Find observability Grammian $\mathbf{G}_o$:
      $\mathbf{T}^T\mathbf{G}_o + \mathbf{G}_o\mathbf{T} = -\mathbf{C}^T\mathbf{C}$;
3 Compute Cholesky factors of $\mathbf{G}_c$ and $\mathbf{G}_o$:
      $\mathbf{G}_c = \mathbf{L}_c\mathbf{L}_c^T, \quad \mathbf{G}_o = \mathbf{L}_o\mathbf{L}_o^T$;
4 Compute SVD of Cholesky product:
      $\mathbf{U}\mathbf{\Sigma}\mathbf{V} = \mathbf{L}_o\mathbf{L}_c^T$
5 Compute transformation matrices $\mathbf{W}_L$ and $\mathbf{W}_R$:
      $\mathbf{W}_L = \mathbf{L}_c\mathbf{V}\mathbf{\Sigma}^{-1/2}$ , $\mathbf{W}_R = \mathbf{\Sigma}^{-1/2}\mathbf{U}^T\mathbf{L}_o^T$;
```

TBR is applicable to linear systems and hence suitable for the pressure equation (2.5). It provides guaranteed stability and improved accuracy over POD with a-priori error bound. However, it is limited to models with few hundred unknowns as the computational complexity is $O(n^3)$. In order to allow effective and efficient reduction for large models, we employ a two-stage reduction strategy where we perform an intermediate reduction using POD method to reduce the model from n to q, and then apply a TBR-based projection around initial stable point (p_0, s_0) to further reduce the model from q to ℓ. The two-stage reduction is summarized in Algorithm 2. The dimensions of $\mathbf{\Phi}_L$ and $\mathbf{\Phi}_R$ are $\ell \times n$ and $n \times \ell$ respectively. The projection matrices therefore reduce the number of unknowns from n to ℓ, with $\ell << n$. The final reduced pressure equation becomes:

$$\tilde{\mathbf{T}}(S^n)\tilde{\mathbf{p}}^{n+1} = \tilde{\mathbf{B}}\mathbf{u}^{n+1} + \tilde{\boldsymbol{G}}^{n+1}, \tag{2.38}$$

where $\tilde{\mathbf{T}} = \mathbf{\Phi}_L^T \mathbf{T} \mathbf{\Phi}_R \in \mathbb{R}^{\ell \times \ell}$ is the reduced transmissibility matrix, $\tilde{\mathbf{p}} \in \mathbb{R}^{\ell \times 1}$ is the reduced pressure vector we seek to solve, $\tilde{\mathbf{B}} = \mathbf{\Phi}_L^T \mathbf{B} \in \mathbb{R}^{\ell \times m}$ is the reduced control arrangement matrix, $\mathbf{u}^{n+1} \in \mathbb{R}^{m \times 1}$ is the input controls, and $\tilde{\boldsymbol{G}} \in \mathbb{R}^{m \times 1}$ is the reduced gravitational force vector. Finally, the pressure state vector $\mathbf{P}$ is reconstructed using (2.13).

Algorithm 2: Two-stage reduction algorithm

```
input  : Pressure snapshot matrix $\hat{\mathbb{P}}$
output: Projection bases $\mathbf{\Phi}_L$ and $\mathbf{\Phi}_R$
1 Compute first-stage reduction basis $\mathbf{U}$ using POD:
      $\mathbf{U\Sigma V} = \mathbb{P}$;     where $\mathbf{U} = n \times$ q;
2 Compute second-stage basis $\mathbf{W}_L$, $\mathbf{W}_R$ using TBR:
      $[\mathbf{W}_L, \mathbf{W}_R] = \mathrm{TBR}(\hat{\mathbf{T}}, \hat{\mathbf{B}}, \hat{\mathbf{C}})$
      where $\hat{\mathbf{T}} = \mathbf{U}^T\mathbf{TU}$,  $\hat{\mathbf{B}} = \mathbf{U}^T\mathbf{B}$,  $\hat{\mathbf{C}} = \mathbf{I}$;
3 Construct final bases:
      $\mathbf{\Phi}_L = \mathbf{W}_L^T\mathbf{U}^T$,   $\mathbf{\Phi}_R = \mathbf{UW}_R$;
```

2.2.2 Reduced-Order Saturation Equation

Nevertheless, linear reduction techniques reduce the dimensions of spatial models in the sense that far fewer variables are present, they still depend on the dimension of the original full-order model—through nonlinear terms—and thus marginal computational gain is achieved. To see this, lets consider reducing the residual and Jacobian of (2.5) by representing the saturation state vector $\mathbf{S}$ in terms of a reduced state vector $\mathbf{S}_r$ using a basis matrix $\boldsymbol{\Upsilon}$ obtained from a saturation snapshot matrix $\mathbb{S}$, as follows:

$$\mathbf{S} = \boldsymbol{\Upsilon}\mathbf{S}_r, \tag{2.39}$$

where $\boldsymbol{\Upsilon} \in \mathbb{R}^{n\times\ell}$ is the saturation basis matrix, and $\mathbf{S}_r \in \mathbb{R}^{\ell\times 1}$ is the reduced saturation vector. Substituting (2.39) into (2.11) and (2.12) and premultiplying by $\boldsymbol{\Upsilon}^T$, the reduced forms of the residual and Jacobian become:

$$\mathbf{g}_r = \boldsymbol{\Upsilon}^T\mathbf{g} = \mathbf{S}_r^{n+1} - \mathbf{S}_r^n - \frac{\Delta t}{\phi}\boldsymbol{\Upsilon}^T\left[\mathbf{F}^{n+1} + \mathbf{Q}_w\right], \tag{2.40}$$

$$\mathbf{J}_r = \boldsymbol{\Upsilon}^T\mathbf{J}\boldsymbol{\Upsilon} = \mathbf{I}_r - \frac{\Delta t}{\phi}\boldsymbol{\Upsilon}^T\left(\frac{\partial \mathbf{F}^{n+1}}{\partial \boldsymbol{\Upsilon}\mathbf{S}_r^{n+1}}\right)\boldsymbol{\Upsilon}, \tag{2.41}$$

where $\mathbf{F}^{n+1} \equiv \mathbf{F}(\boldsymbol{\Upsilon}\mathbf{S}_r^{n+1})$ is the reduced flux vector and $\mathbf{I}_r$ is the identity matrix at the reduced dimension. It is clear that the evaluation of the flux vector still requires $O(n)$ nonlinear evaluations for the n entries of $\mathbf{F}(\boldsymbol{\Upsilon}\mathbf{S}_r^{n+1})$ and $\mathbf{F}'(\boldsymbol{\Upsilon}\mathbf{S}_r^{n+1})$ which is proportional to the dimension of the FOM. Specifically, the flux term is evaluated at the full dimension first before projection onto the reduced dimension.

To avoid full evaluation of the flux term, DEIM is proposed to approximate the nonlinear flux term via interpolation over a subset of points that are independent of the FOM dimension n. In this context, DEIM partially evaluates the flux term at *"interpolation indices"* which represent in this problem the grid blocks that are essential in reconstructing the flux term while preserving the overall properties and continuity of the saturation equation (e.g., Figure 2.1).

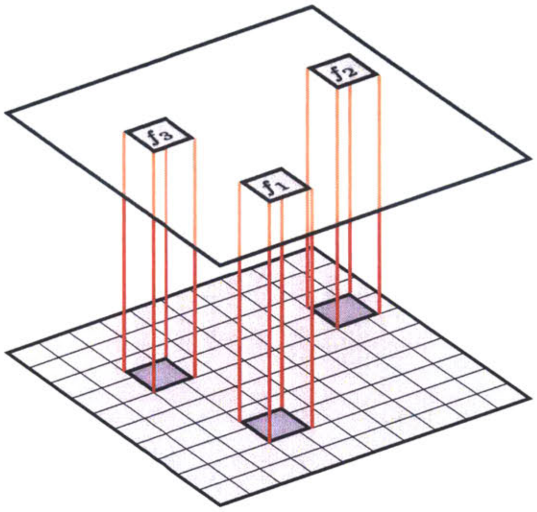


Figure 2.1: Representation of a spatially discretized nonlinear function f using three interpolation points. The top plane represents the actual function evaluation at three grid blocks while the bottom plane represents the reconstructed function using DEIM.

The first step of the DEIM procedure is to approximate the flux term $\mathbf{F}^{n+1}$ using a separate set of basis vectors $\mathbf{\Psi}$ that are different from those used for the states:

$$\mathbf{F}^{n+1} \approx \mathbf{\Psi}\mathbf{F}_r^{n+1}, \tag{2.42}$$

where $\mathbf{\Psi} = [\psi_1, \psi_2, \cdots, \psi_k] \in \mathbb{R}^{n \times k}$ $(k \ll n)$ are the basis vectors for the flux term and $\mathbf{F}_r^{n+1} \in \mathbb{R}^{k \times 1}$ is the vector containing expansion coefficients. The basis vectors $\mathbf{\Psi} = \{\psi_i\}_{i=1}^{k}$ are derived from the snapshots of the flux term $\mathbb{F} = \{\mathbf{F}_i\}_{i=1}^{n_f}$, using the same POD procedure described earlier. The collection of these nonlinear snapshots does not incur additional computational cost because the nonlinear flux term is already evaluated during the sampling of the state snapshots $\mathbb{P}$ and $\mathbb{S}$. Note that (2.42) represents an overdetermined system, so to compute $\mathbf{F}_r^{n+1}$ at a computational cost independent of n, we select k interpolation points of $\mathbf{F}^{n+1}$ and enforce equality for the corresponding system of equations in (2.42). This results in a $k \times k$ system from which the coefficient vector $\mathbf{F}_r^{n+1}$ can be uniquely determined:

$$\mathbf{F}_{\vec{\mathbf{z}}}^{n+1} = \mathbf{\Psi}_{\vec{\mathbf{z}}}\mathbf{F}_r^{n+1}, \tag{2.43}$$

where $\vec{\mathbf{z}} = [z_1, \cdots, z_k]^T \in \mathbb{R}^{k\times 1}$ is a vector containing k interpolation indices that are determined inductively via Algorithm 3, $\mathbf{F}_{\vec{\mathbf{z}}}^{n+1} \in \mathbb{R}^{k\times 1}$ is the nonlinear flux term evaluated at these k interpolation points, and $\mathbf{\Psi}_{\vec{\mathbf{z}}} \in {}^{k\times k}$ is the corresponding k rows of $\mathbf{\Psi}$.

For problems where nonlinear terms have componentwise dependence on the state (e.g., two-phase flow with no gravitational effects), each grid block evaluation of the nonlinear term is independent of the neighboring grid blocks and therefore can be efficiently reconstructed using only k interpolation points. However, the problem considered in this work includes gravitational effects, which transform the system from a componentwise to a non-componentwise dependence of the nonlinear flux term on the solution states. That is, each grid block evaluation of the flux term requires evaluating the neighboring grid blocks due to transmissibility upstream weighting and therefore the flux vector is reconstructed using grid blocks selected via interpolation indices in addition to those neighboring grids selected according to upstream weighting (2.8) . This is illustrated in Figure 2.2 for a two-dimensional model. A total of 25 grid blocks are used to reconstruct the nonlinear function f—five of which are interpolation points selected using Algorithm 3. We denote $\vec{\mathbf{z}}'$ as the set of $k' \geq k$ indices which include the grid blocks selected by the DEIM algorithm in addition to the neighboring grid blocks, i.e., $\vec{\mathbf{z}} \in \vec{\mathbf{z}}'$. Even though, an increase in computational demands is anticipated in this case, the support of states on the flux term remains local, i.e., $k \leq k' \ll n$.

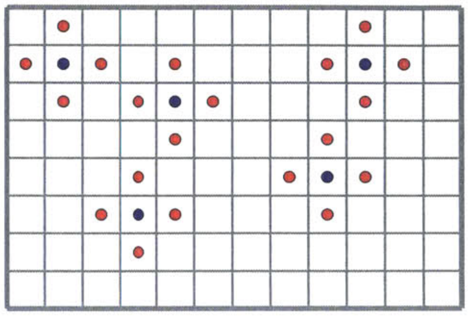


Figure 2.2: Illustration of a spatially discretized nonlinear function f with non-componentwise dependance. The blue circles indicate the grid blocks selected by Algorithm 3 whereas the red circles are the neighboring grid blocks required due to upstream weighting to reconstruct the function.

Solving for $\mathbf{F}_r^{n+1}$ from (2.43) and substituting into (2.42), we obtain the final DEIM approximation:

$$\mathbf{F}^{n+1} = \mathbf{\Psi}\mathbf{\Psi}_{\vec{\mathbf{z}}}^{-1}\mathbf{F}_{\vec{\mathbf{z}}}^{n+1}. \tag{2.44}$$

Applying the DEIM approximation in (2.44) to the reduced residual (2.40):

$$\mathbf{g}_r = \mathbf{S}_r^{n+1} - \mathbf{S}_r^{n} - \frac{\Delta t}{\phi}\mathbf{\Upsilon}^T\Big[\mathbf{\Psi}\mathbf{\Psi}_{\vec{\mathbf{z}}}^{-1}\mathbf{F}_{\vec{\mathbf{z}}}^{n+1} + \mathbf{Q}_w\Big]. \tag{2.45}$$

Here $\mathbf{F}_{\vec{\mathbf{z}}}^{n+1} \equiv \mathbf{F}_{\vec{\mathbf{z}}}(\mathbf{\Upsilon}_{\vec{\mathbf{z}}'}\mathbf{S}_r^{n+1})$ for non-componentwise evaluation. Similarly for the Jacobian matrix in (2.12):

$$\hat{\mathbf{J}} = \mathbf{I}_r - \frac{\Delta t}{\phi}\mathbf{\Upsilon}^T\mathbf{\Psi}\mathbf{\Psi}_{\vec{\mathbf{z}}}^{-1}\left(\frac{\partial \mathbf{F}_{\vec{\mathbf{z}}}^{n+1}}{\partial \mathbf{\Upsilon}_{\vec{\mathbf{z}}'}\mathbf{S}_r^{n+1}}\right)\mathbf{\Upsilon}_{\vec{\mathbf{z}}'}, \tag{2.46}$$

where $\mathbf{\Upsilon}_{\vec{\mathbf{z}}'} \in \mathbb{R}^{k' \times \ell}$ contains the corresponding k' rows of $\mathbf{\Upsilon}$ and $\left(\frac{\partial \mathbf{F}_{\vec{\mathbf{z}}}^{n+1}}{\partial \mathbf{\Upsilon}_{\vec{\mathbf{z}}'}\mathbf{S}_r^{n+1}}\right) \in \mathbb{R}^{k \times k'}$.

The computation of the reduced residual and Jacobian in (2.45) and (2.46) is now efficient because firstly, the term $\boldsymbol{\Upsilon}^T\boldsymbol{\Psi}\boldsymbol{\Psi}_{\vec{\mathbf{z}}}^{-1} \in \mathbb{R}^{\ell\times k}$ can be precomputed *"offline"* meaning that it is computed only during training procedure and thus does not need to be computed within Newton-Raphson iterations or any multi-query context. Secondly, the flux term $\mathbf{F}_{\vec{\mathbf{z}}}^{n+1}$ does not entail the full-order dimension n anymore as it can be obtained by extracting rows of $\mathbf{F}^{n+1}$ corresponding to $\vec{\mathbf{z}}'$ and it is therefore $O(k')$.

```
Algorithm 3: DEIM Point Selection
Input  : $\boldsymbol{\Psi} = \{\psi_i\}_{i=1}^{k}$
Output: $\vec{\mathbf{z}} = [z_1, z_2, \cdots, z_k] \in \mathbb{R}^{k\times 1}$
1 $[|\rho|, z_1] = \max\{|\psi_1|\}$
2 $\boldsymbol{\Psi} = [\psi_1], \mathbf{K} = [\mathbf{e}_{z_1}], \vec{\mathbf{z}} = [z_1]$
3 for $i \leftarrow 2$ to $M$ do
      Solve $(\mathbf{K}^T\boldsymbol{\Psi})\mathbf{c} = \mathbf{K}^T\psi_i$ for $\mathbf{c}$
      $\mathbf{r} = \psi_i - \boldsymbol{\Psi}\mathbf{c}$
      $[|\rho|, z_i] = \max\{|\mathbf{r}|\}$
      $\boldsymbol{\Psi} \leftarrow [\boldsymbol{\Psi}, \psi_i]$ , $\mathbf{K} \leftarrow [\mathbf{K}\ \ \mathbf{e}_{z_i}], \vec{\mathbf{z}} = [\vec{\mathbf{z}}^T, z_i]$
  end
```

2.3 Simulation Using Reduced-Order Modeling

We now illustrate the application of the ROM-based procedure on a synthetic model containing 13,200 grid blocks. The performance of ROM is tested and verified for a variety of test controls that differ from initial controls used during the initial training simulation to demonstrate the ability of the ROM in prediction mode. Simulation runtimes and speedup factors are quantified at the end of the section.

The reservoir model represents the first layer of the SPE 10th comparative study [34].

The grid is 60 × 220 × 1 ($N_x \times N_y \times N_z$, where N_k is the number of grid blocks in direction k). The physical dimensions of each grid block is 20 ft × 10 ft × 2 ft. There are four production wells located at the corners of the model and one injection well located at the center. All of the wells are under flow rate control.

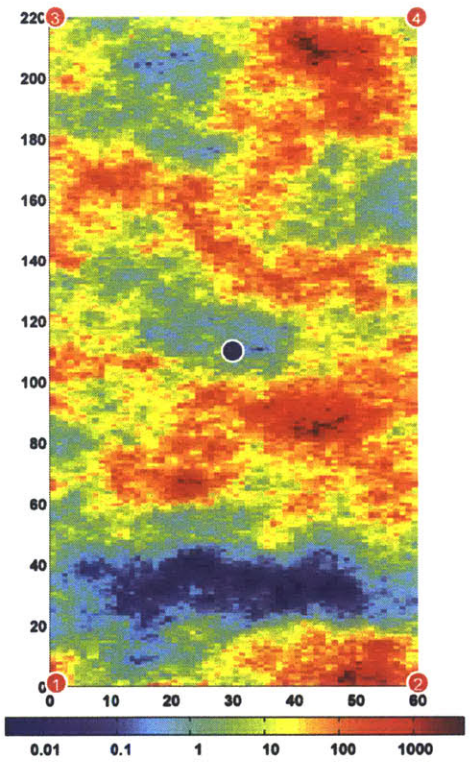


Figure 2.3: Permeability in the x-direction (in mD) of the first layer of the SPE 10th comparative model with four production wells placed at the corners and an injection well placed in the middle.

Permeability in the x-direction is depicted in Figure 2.3. Permeability is taken to be a diagonal tensor, with $k_x = k_y$. The mean k_x and k_z are 74.3 mD and 17.1 mD respectively. The porosity is constant in this model and set equal to 0.25. The initial water and residual oil saturations are zero. For oil, we set $\rho_o = 45$ lb/ft^3, $\mu_o = 5$ cp; for water, we set $\rho_w = 65$ lb/ft^3, $\mu_w = 1$ cp. The system is incompressible and capillary pressure and gravity effects are neglected. The relative permeabilities for the oil and water phases are specified as:

$$k_{ro}(S_w) = k_{ro}^0 \left(\frac{1 - S_w - S_{or}}{1 - S_{wr} - S_{or}} \right)^a, \tag{2.47}$$

$$k_{rw}(S_w) = k_{rw}^0 \left(\frac{S_w - S_{or}}{1 - S_{wr} - S_{or}} \right)^b, \tag{2.48}$$

where k_{ro}^0 and k_{rw}^0 are the endpoint relative permeabilities. Here we set $k_{ro}^0 = k_{rw}^0 = 1$ and $a = b = 2$.

We perform a training simulation run using the FOM to collect state and nonlinear term snapshots in order to construct the orthogonal bases. The flow rate control schedule is shown in Figure 2.4. We specify the injection well to inject water at a constant rate of 211 STB/day, equivalent to 0.6 of the reservoir pore volume (0.6 PVI), while the production wells produce also constantly at 53 STB/day. The training case was simulated for 1000 days using MRST, and a total of 200 snapshots are collected and used to build the orthogonal bases.

Applying the proper orthogonal decomposition approach provides the amount of omitted energy for each state as a function of the number of retained POD bases shown in Figure 2.5. We show here the first 40 orthogonal bases out of the 200 available. It is clear that retaining only the first pressure basis results in omitting 1×10^{-4} of the energy of the pressure system. However, as indicated earlier and for this exercise, we find that more pressure bases are required to achieve accurate results. This is due to the fact that slight perturbation in pressure, which is represented by orthogonal basis with low energy, is critical to the pressure-dependent terms supplied to the saturation equation. A slight change in these pressure-dependent terms may result in the saturation equation converging to entirely wrong solutions during Newton iterations.

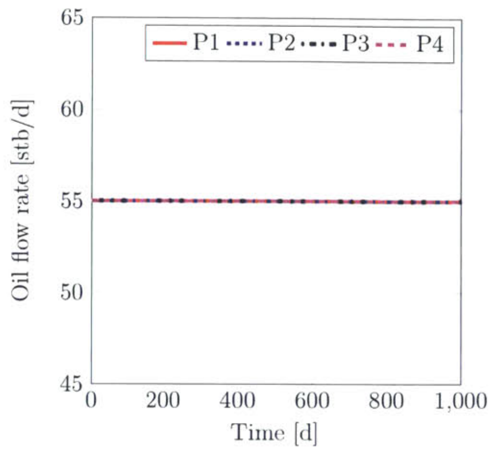


Figure 2.4: Training flow rate schedule for the first layer of the SPE 10th comparative model. The training schedule consists of all production wells production at a constant flow rate of 55 STB/day.

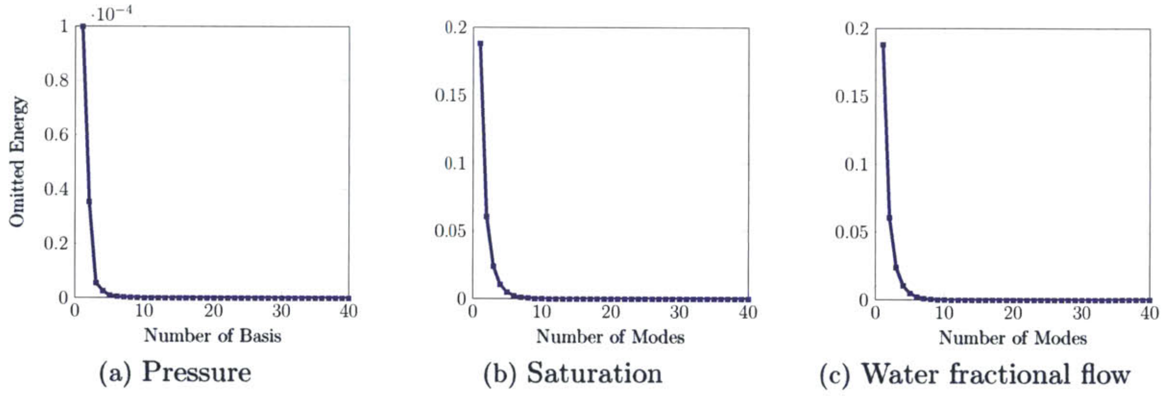


Figure 2.5: Omitted Energy as a function of the number of POD bases. The omitted energy is used in this example to determine the number of retained basis.

Therefore, a total of 70 basis vectors are retained comprising less than 1×10^{-10} of the omitted energy. Then TBR is used to reduced the number of basis vectors further to 25 basis vectors. Applying TBR does not only result in more stable basis matrix, but also helps in retaining the accuracy of the ROM while keeping the ROM at lower dimension. The saturation and fractional flow basis matrices are reduced using POD and contain 16 and 15 basis vectors respectively with approximately 1×10^{-6} of the energy omitted . We thus reduce the dimension of the problem from 26,400 variables to only 41 variables. The nonlinear term, water fractional flow, orthogonal basis matrix is used to determine the interpolation indices for DEIM. The spatial location of each DEIM point is depicted in Figure 2.6. The DEIM points are concentrated in locations where the rate of change in flux are high, i.e., the path toward well P2. Although DEIM points corresponding to all snapshots are shown in Figure 2.6, only the red DEIM points (15 points) are used to recover the water fractional flow term. Thus, the whole saturation field, evaluated using the saturation equation, can be recovered accurately by evaluating the fractional term at only those locations.

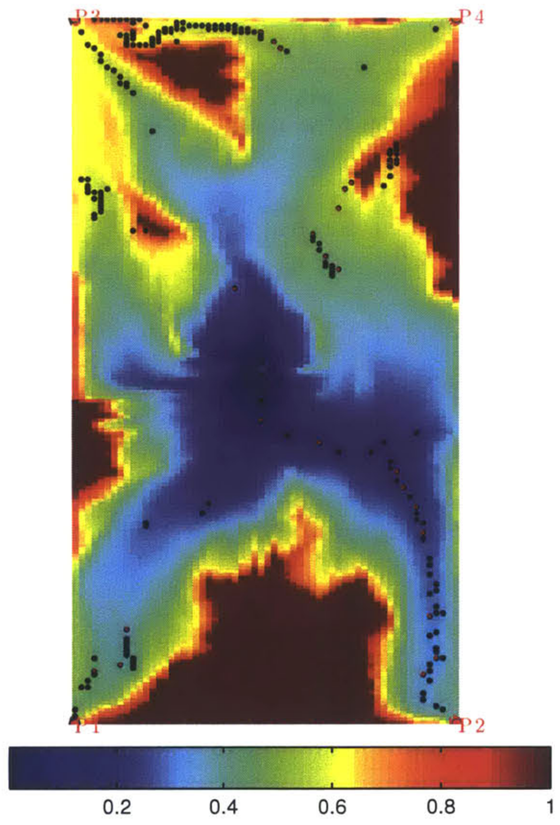


Figure 2.6: Oil saturation at the final time step with DEIM points for the flux term of the first layer of the SPE 10th comparative model. The DEIM points are used to reconstruct the nonlinear fractional flow term and are located relatively close to production wells.

Prior to using the ROM for optimization and decision-making, it is important to assess the accuracy of the constructed ROM. We follow He et al. [54] testing methodology where we define a *'target'* flow rate schedule for each well. The flow rate control for each well in the target flow rate schedule, depicted in Figure 2.7, is randomly set within an interval between 43 STB/day and 74 STB/day under the constraint that the injected fluid (water) has to be equal to the produced fluid (oil and water) for incompressible flow model. The target flow rate schedule for each well is perturbed every 100 days. Therefore, the total number of control variables is $10 \times 5 = 50$. It is clear that the training flow rate schedule, taken to be constant for all wells, is vastly different from the target flow rate schedule. We then interpolate between the training and target flow rates to enable a systematic perturbation away from the training run. Specifically, we specify the test case flow rates as follows:

$$\mathbf{u}_{\text{test}} = (1-\alpha)\mathbf{u}_{\text{training}} + \alpha\mathbf{u}_{\text{target}}, \tag{2.49}$$

where α is taken to be between 0 and 1. The error between the FOM and ROM is expected to increase with increasing α as the test case is entirely made from the target flow rate schedule when α is 1.

The ROM performance is scrutinized using three test cases. Figure 2.8 shows the the oil and water production rates for the four production wells for Test 1. In this and subsequent figures, the dashed lines show the training simulation results which controls were used to build the ROM, the solid lines display the reference test case results simulated using the FOM, and the circles depict the test case results simulated using the ROM. Test 1 is simulated while setting α equal to zero indicating that testing rate schedule is similar to that used for training. The results show full agreement between the FOM and the ROM when similar controls are used. For Test 2, we set α equal to 0.5. The oil and water production comparisons are shown in Figure 2.9.

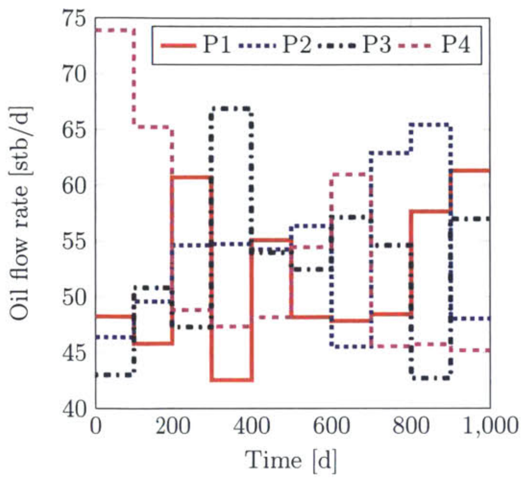


Figure 2.7: Testing flow rate schedule for the first layer of the SPE 10th comparative model. Production flow rates are perturbed every 100 days and randomly set within an interval between 43 STB/day and 74 STB/day. The testing schedule is used to test the ROM's performance and solution accuracy.

ROM results are in close agreement with the FOM for all quantities. We see that the test simulation results are slightly different from the training results, though there are conspicuous differences between the two cases particularly for the oil production in wells P3 and P4. In Test 3, we specify α to be equal to one, which means we apply the flow rate schedule shown in Figure 2.7. Results for this case are shown in Figure 2.10. The ROM results are slightly less accurate than Test 2 which is expected as we deviate away from the ROM training root-point. Both trajectories are however still in close overall agreement with the FOM. We notice less agreement for wells P1 and P2 toward the end of the simulation run as water production increases. This might be due to inadequate DEIM points and so the error can be rectified by increasing the DEIM points. Overall, the results for all test cases demonstrate that the ROM is able to provide simulation results that are in close agreement to those from the FOM. For this reservoir model, the full-order simulation runtime is about 280 seconds, while the runtime for the ROM is about 3 seconds. We thus achieve a reduction of computational time by a factor of 93. Although this reduction in computation is considered modest as compared to computation time reduction of $O(1000)$ attained by [33] when using DEIM for nonlinear miscible viscous fingering problem, we anticipate larger speedup factors when larger models are used ($O(10^5)$ grid blocks). This is firstly because TBR reduction will be more significant (current reduction factor for this model is 528 for pressure equation). Secondly, DEIM eliminates full evaluation of nonlinear terms and therefore less time will be spent during the Newton-Raphson iterations which will be very advantageous for large models with large number of variables.

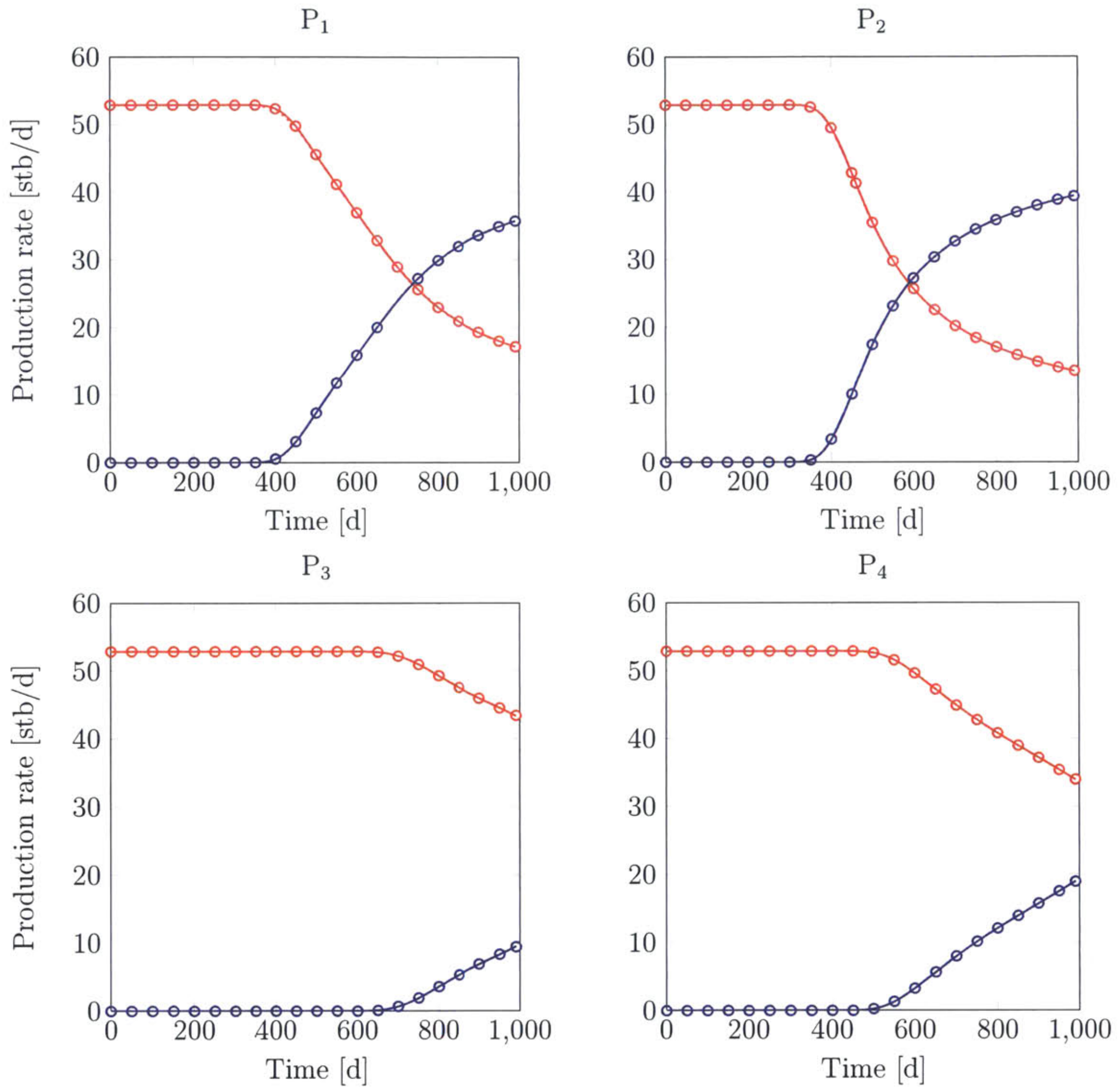


Figure 2.8: Production rates for Test 1 ($\alpha = 0$). The dashed lines show the training simulation results which controls were used to build the ROM, the solid lines display the reference test case results simulated using the FOM, and the circles depict the test case results simulated using the ROM.

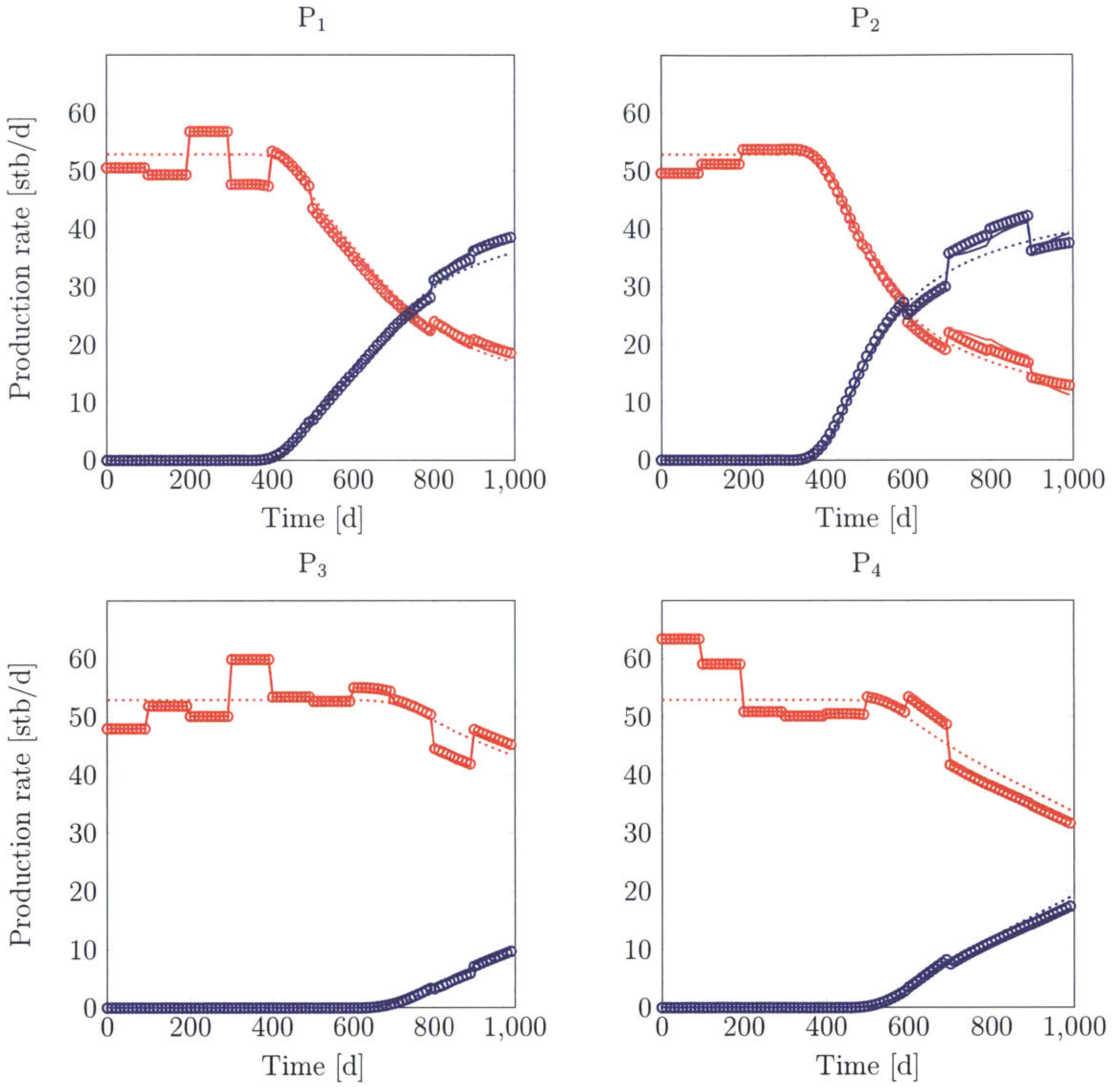


Figure 2.9: Production rates for Test 2 ($\alpha = 0.5$). The dashed lines show the training simulation results which controls were used to build the ROM, the solid lines display the reference test case results simulated using the FOM, and the circles depict the test case results simulated using the ROM.

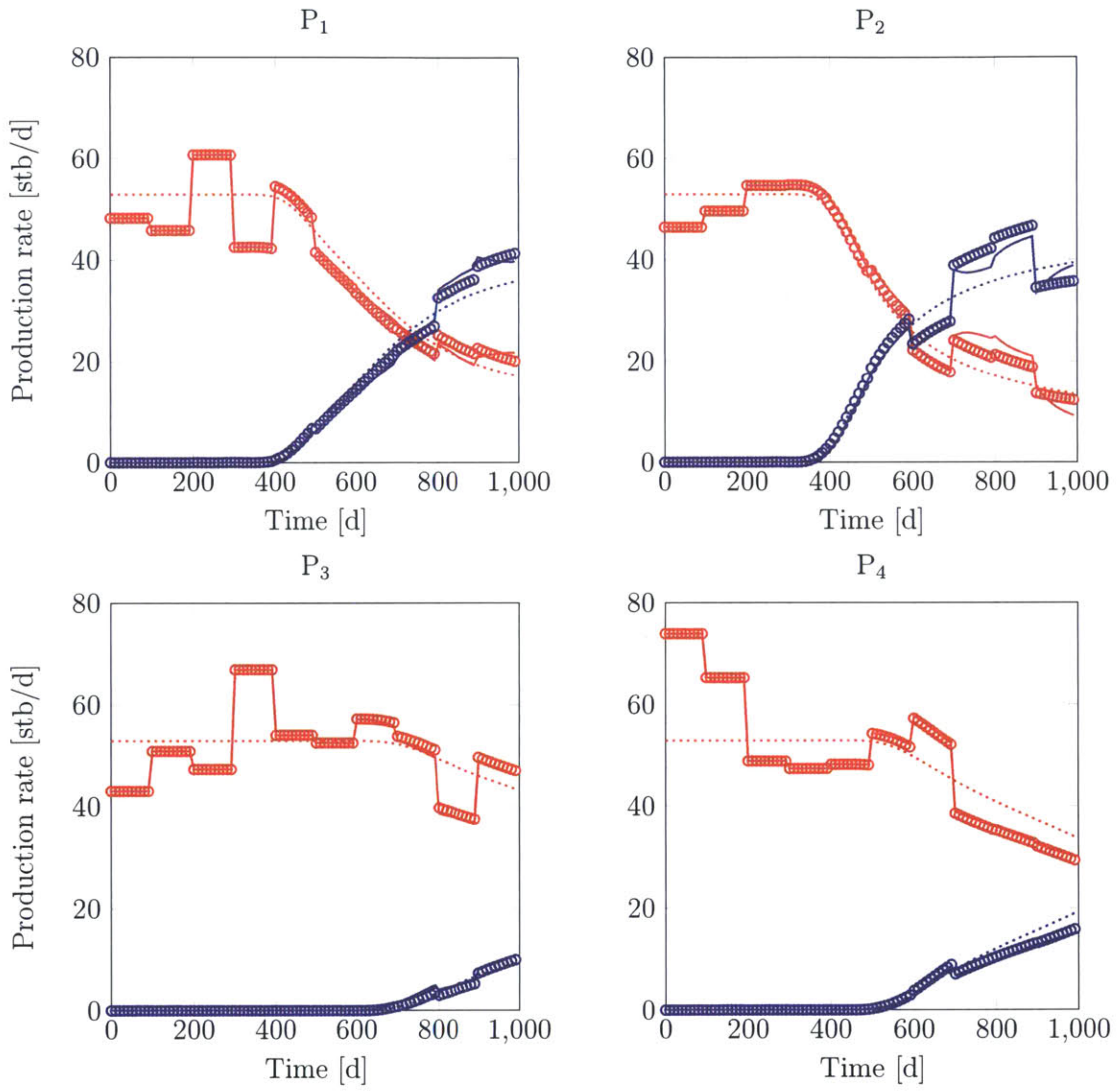


Figure 2.10: Production rates for Test 3 ($\alpha = 1.0$). The dashed lines show the training simulation results which controls were used to build the ROM, the solid lines display the reference test case results simulated using the FOM, and the circles depict the test case results simulated using the ROM.

2.4 Summary

In this chapter, we developed a reduced-order modeling technique for reservoir management decision-making and optimization. The technique entails combining proper orthogonal decomposition (POD) and truncated balanced realization (TBR) for model-order reduction and using discrete empirical interpolation (DEIM) for nonlinear term approximation. A high degree of efficiency is achieved as far fewer variables are required to reproduced the FOM input/output behavior. DEIM improves on the shortcomings of the original projection-based model-order reduction techniques by intelligently approximating nonlinear parameters depicted in the saturation equation.

The ROM technique was applied to a heterogenous two-dimensional model containing 13,200 grid blocks and five wells. The accuracy of the ROM was demonstrated for a sequence of testing runs. Production trajectories were shown to be in close agreement to those computed using FOM. A computational speedup factor of about 93 was achieved and we expect higher factors for larger models as we will see in subsequent chapters.

One primary application of ROM is decision-making and optimization. In the following chapter, we will develop an optimization framework that utilizes the developed ROMs to solve the well placement and control problems. The framework provides the ability to retrain and update the ROM during the optimization to ensure highest solution accuracy.

Chapter 3

Optimization of Well Placements and Controls Using Reduced-Order Models

Our goal in this chapter is to develop an efficient well placement and control optimization techniques that take advantage of the developed reduced-order model approach. Section 3.1 presents the well placement and control problems in their mathematical forms and highlights the nomenclature used throughout the chapter. As stated previously, the well placement problem is multi-modal with multiple local optima. Therefore in Section 3.2, we present a new global optimization method known as Modified Cuckoo Search (MCS). MCS which was originally developed for unconstrained problems is enhanced in this work to handle nonlinear constraints via filtering. Section 3.3 describes a local optimization method known as Mesh Adaptive Direct Search (MADS) that is coupled with MCS to formulate a new hybrid approach. This hybrid approach, known as MCS-MADS, combines the positive aspects of the two preceding methods and is shown to provide better solution quality. The hybrid approach constitutes an adaptive framework that allows for the use of ROM as a search accelerator to reduce computational demands. The framework also includes error assessment procedure to ensure minimum deviation from the reference FOM. Section 3.5 addresses the well control problem. Controls are optimized using a gradient optimization method where gradients are obtained through adjoint procedures. ROMs are used in this problem as the main engine driving the optimizer toward the optimal solution. To keep the ROM close to its root-point, optimization is performed within a trust-region. The quality of the optimal solution obtained using the ROM is compared to that of the reference FOM and adjustment to the next trust-region is performed accordingly.

3.1 Problem Statement

$$\begin{aligned} \underset{\mathbf{u}\in U}{\text{minimize}} \quad & J(\mathbf{p},\mathbf{u}) \\ \text{subject to} \quad & \mathbf{c}(\mathbf{p},\mathbf{u}) \le 0 \\ & \mathbf{g}(\mathbf{p},\mathbf{u}) = 0 \end{aligned} \tag{3.1}$$

where J is the objective function to be optimized (e.g., net present value, recovery factor), $\mathbf{c} \in \mathbb{R}^k$ is the set of k nonlinear constraint functions (e.g., water cut, maximum liquid field production). The equality constraint $\mathbf{g}$ defines both the bound constraints (i.e., $\mathbf{L}_{\text{B}} \le \mathbf{u} \le \mathbf{U}_{\text{B}}$) and the dynamic system constraint, which is the reservoir simulator in this case. This constraint ensures that the governing equations for flow converge toward the correct solution for each proposed control $\mathbf{u}$ (essential condition for adjoint-based optimization techniques). For all examples in this work, we use MATLAB Reservoir Simulation Toolbox (MRST). Readers should refer to [69] for details about MRST. The vector $\mathbf{p}$ represents the pair of the simulation model solution states (i.e., pressure p and saturation S for the case of two-phase flow) and $\mathbf{u}$ represents the set of input controls we wish to find in order to optimize the objective function, (e.g., well drilling locations or well control settings).

In this work, we consider a sequential approach where we optimize the well placements first before finding the optimal set of well controls. Since the problem cannot be entirely decoupled—optimal well locations still require well control settings during the optimization—we apply a reactive approach where wells are shut-in if the water production exceeds a specified limit. This approach is shown to provide the best results for such problems [112].

The control vector $\mathbf{u}$ is defined differently for each problem. For the well placement problem, the input control vector $\mathbf{u}$ is defined as the set of discrete integer values representing the potential well locations indexed using (i, j). Its bounds include the entire connected set of the model's grid blocks as in the case of new field development or the non-connected feasible regions in the case of infill drilling where a minimum drainage area is usually imposed. For the well control problem, the input control $\mathbf{u}$ represents the continuous well flow rate or BHP. It is bounded by operational limits (e.g., for BHP control, BHP should be higher than the bubble point pressure at producers and less than fracture pressure at injectors).

The well placement problem is multi-modal (contains local optima) due to the effect of reservoir heterogeneity—normally displays non-smooth optimization surface. Therefore, a global optimization method with local search capability, MCS-MADS, is devised to avoid getting trapped in any local optimum. On the other hand, the well control problem displays smoother optimization surface and therefore gradient-based algorithms are most suitable. In this work, we apply an adaptive trust-region framework that incorporates the use of ROMs to optimize the well controls.

3.2 Global Derivative-Free Approach for Well Placements

3.2.1 Cuckoo Search

Cuckoo Search algorithm (CS), introduced by Yang and Deb [106], is one of the newly developed global metaheuristic search procedures. It is inspired by breeding behavior such as brood parasitism that is displayed by certain species of cuckoos. Brood parasites (such as cuckoo birds) manipulate and use other individuals, either of the same or different species, to raise their own young while the parasitic parent pursues other activities like foraging or producing offspring. This interesting behavior is combined with a Lèvy flight approach—exhibited by some birds and fruit flies—to efficiently search the solution space. Lévy flight allows exploring the search space using a series of straight flight paths punctuated by sudden sharp turns (close to $90°$) to perform intermittent free search pattern.

CS entails the use of eggs to represent search agents and nests to hold these agents during the optimization. Adopting a similar analogy to that presented in [102], the host bird egg originally placed in a nest represents a solution and the cuckoo egg represents a new potential solution. Each egg carries two pieces of information, that is, the solution which is represented as coordinates in the solution space and its fitness value. The nests act as place holders for the eggs, and can be thought of as locations in an array where eggs information is stored during the optimization run. Each nest hosts one egg at a time and the number of nests is fixed throughout the optimization (similar to population size in genetic algorithms).

The CS algorithm starts by generating an initial population of eggs, this is equivalent to the host birds building nests and laying one egg in each nest. The initial population should have an adequate number of eggs to provide efficient sampling of the solution space. Latin hyper-cube sampling technique (LHS) is used here for this purpose. During subsequent generations, cuckoo eggs are generated by taking a Lèvy flight from a random host egg (or previous cuckoo egg). The fitness of each cuckoo egg is evaluated, and compared to a random previous egg (either host or cuckoo). If the fitness of the new cuckoo egg is better than the previous egg, the cuckoo egg replaces that egg in the nest.

Although cuckoo birds cleverly disguise their eggs (they choose nests that contain eggs of similar appearance to theirs), there is always a probability $p_a \in [0, 1]$ that the host bird discovers the cuckoo's eggs inside its nest. Depending on the type of the host bird, it may discard those cuckoo eggs, or abandon the nest altogether and build a new one. The CS algorithm emulates this behavior by discovering the worst eggs with a probability p_a and replacing them with new eggs using a biased random walk approach. This rule ensures that best eggs survive from generation to generation with the best eggs always carried over (elitism).

One of the most distinctive features of CS when compared to other metaheuristic techniques is the use of Lèvy flight to generate new eggs. It is a random walk, characterized by a series of instantaneous jumps generated by a probability density function as illustrated in Figure 3.1. Lèvy flight approach constitutes a scale-free search strategy meaning that the search pattern is the same regardless of the scale of the problem. This provides an automatic balance between exploration (global search) and exploitation (local search) with only one parameter to tune.

Another major strength of CS algorithm is its simplicity. While other metaheuristic algorithms such as GA or PSO require a handful of problem-dependent parameters to be adjusted, the CS algorithm requires only two, i.e., the fraction of eggs to be

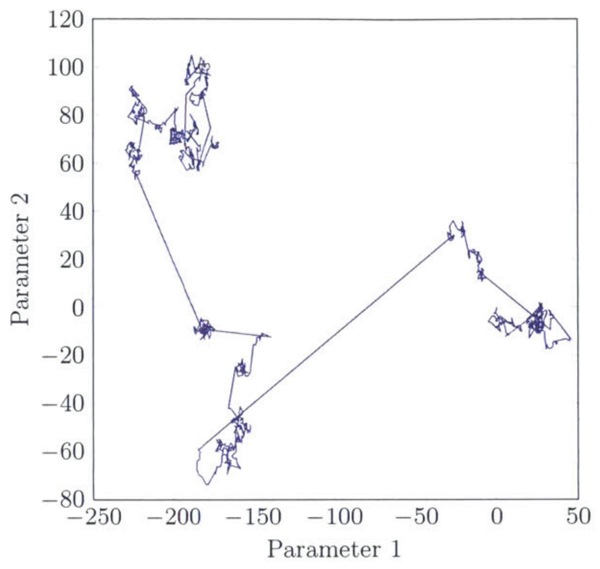


Figure 3.1: An example of a 1000 step Lèvy flight showing small localized searches connected by large jumps. The large jumps are resulted from using a probability distribution that is heavy-tailed.

abandoned (or discovered) during each generation and the Lèvy flight step size. In this work, we use discovery probability recommended by Yang and Deb [106] ($p_a = 0.25$), which was shown to perform well for a suite of test problems.

3.2.2 Modified Cuckoo Search

The CS algorithm has shown to outperform a number of other metaheuristic algorithms [106, 109, 103]. Given adequate number of search agents (eggs), the CS algorithm will always find the global optimal solution. As the search, however, relies entirely on Lèvy flights (random walks), fast convergence cannot be guaranteed. For problems with expensive objective function evaluations, such as fluid flow problems, slow convergence may incur prohibitive computational costs. To address this problem, Modified Cuckoo Search (MCS) is introduced.

MCS is implemented with the primary emphasis on speeding up the convergence toward the optimal solution rather than modeling the breeding behavior of cuckoo birds. Hence, two modifications are made to lessen the dependence on random walks without losing the main attractive features of the original CS. The first modification is made to replace the constant Lèvy flight step size α for a varying step size that gradually decreases as the algorithm approaches the optimal solution. The second modification introduces higher level of information exchange among cuckoos through crossover processes.

The first modification is to alter the constant Lèvy flight step size α. In CS, a constant Lèvy flight step size is used and is typically set to $\alpha = 1$. This constant Lèvy flight step size promotes the same level of search aggressiveness regardless of how close the eggs are to the optimal solution. Instead of a constant α, MCS deploys a variable step size that decreases as the number of generation increases, and hence lessen the search aggressiveness as the algorithm converges toward the optimal solution. Similar behavior is demonstrated in PSO, where the inertial constant is reduced to encourage more localized search as the solution gets closer to the optimal one [23].

In MCS, the initial Lèvy flight step size is set to $\alpha = 1$. For subsequent generations, a new Lèvy flight step size is calculated using $\alpha = A/\sqrt{G}$, where G is the generation number and $A = 1$. This varying step size is only performed on those cuckoos that are to be discovered by the host bird.

The second modification, depicted in Figure 3.2, introduces higher level of information exchange between cuckoos through crossover processes. This causes Lèvy flights to depend on previous information from previous generations, rather than performing independent flights. The process can be summarized into the following steps:

1. Eggs are ordered based on their fitness. A fraction of the best eggs is considered

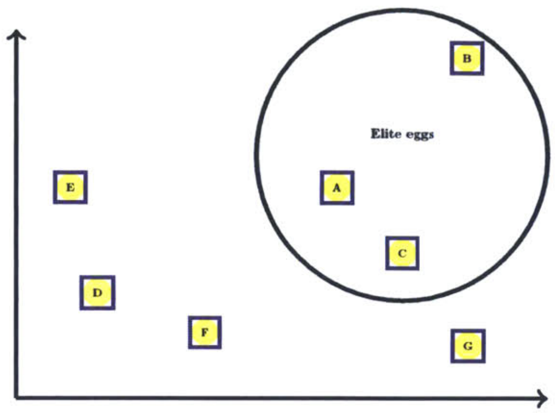
elite (this fraction is determined by the discovery probability p_a), as depicted in Figure 3.2(a).

2. Lèvy flight approach is used to generate new eggs out of non-elite eggs. The non-elite eggs are replaced with the new eggs regardless of whether they incur better fitness as shown in Figure 3.2(b).

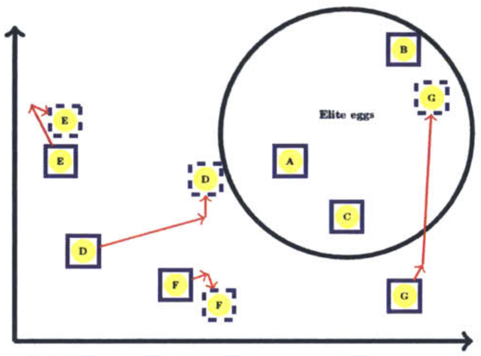
3. Each elite egg randomly picks another elite egg. A new egg is generated along the path connecting these two eggs. The new egg is placed closer to the egg with the better fitness according to the inverse of the golden ratio $\varphi^{-1} = (\sqrt{5}-1)/2$. In the case that both eggs have the same fitness, the new egg is placed in the middle. Also, there is a possibility that an elite egg gets picked twice, in such a case, a local Lèvy flight search is performed from the elite nest (which is picked twice) with a Lèvy flight coefficient $\alpha = A/G^2$.

4. After a new egg is generated using the previous step, it is compared to another egg that is picked randomly (either elite or non-elite). If the newly generated egg has a better fitness than that already in the nest, then it replaces it inside the nest, otherwise it is discarded. This is illustrated in Figure 3.2(c), where egg H is found to have better fitness than egg D, so it replaces it inside the nest (eggs D is left without a nest). On the other hand, egg J is found to have a worst fitness than eggs B so no replacement occurs.

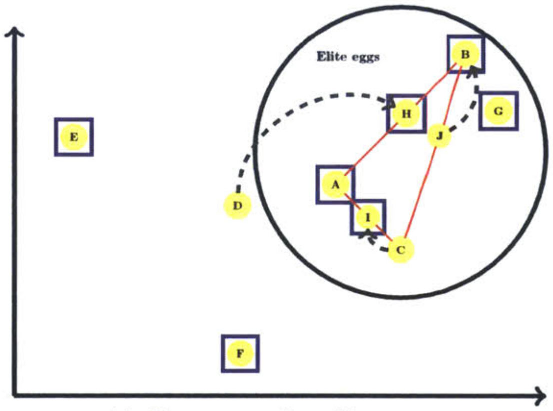
5. All eggs that do not reside inside nests are discarded, and the set of elite eggs is updated as shown in Figure 3.2(d).



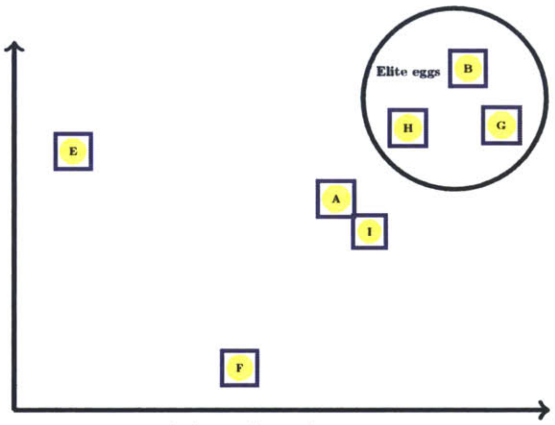
(a) Initial population



(b) Lèvy flights for non-elite eggs



(c) Crossover for elite eggs



(d) updated eggs

Figure 3.2: Illustration of the MCS information exchange among elite eggs with global optimum toward top right (from Walton [102] with modifications).

3.2.3 Constraint Treatment in MCS

Bound constraints in (3.1) are enforced by projecting solutions that fall outside the search domain Ω back onto the boundaries using:

$$\operatorname{proj}_{\Omega}(u_i) = \begin{cases} u_{l_i} & \text{if } u_i < u_{l_i}, \\ u_{u_i} & \text{if } u_i > u_{u_i}, \\ x_i & \text{otherwise.} \end{cases} \tag{3.2}$$

To treat nonlinear constraints, metaheuristic search algorithms commonly apply penalty function approaches which augment the objective function with a penalty term that provides a measurement of constraint violation. This augmented function is referred to as merit function and is expressed as:

$$\chi(\mathbf{p}, \mathbf{u}; \mu) = J(\mathbf{p}, \mathbf{u}) + \wp h(\mathbf{p}, \mathbf{u}), \tag{3.3}$$

where the positive scaler $\wp$ is the penalty parameter, and $h(\mathbf{p}, \mathbf{u})$ is an aggregate constraint violation function expressed as:

$$h(\mathbf{p}, \mathbf{u}) = \left[\sum_{i=1}^{m} (\max(\bar{c}_i(\mathbf{p}, \mathbf{u}), 0))^2\right]^{1/2}, \tag{3.4}$$

$\bar{c}_i(\mathbf{p}, \mathbf{u})$ represents the normalized constraints. Specifically, constraints of the form $c_i(\mathbf{p}, \mathbf{u}) \leq c_{\max}$ are formulated as:

$$\bar{c}_i(\mathbf{p}, \mathbf{u}) = \frac{c_i(\mathbf{p}, \mathbf{u})}{c_{\max}} - 1 \leq 0, \tag{3.5}$$

and those of the form $c_i(\mathbf{p}, \mathbf{u}) \geq c_{\min}$ become:

$$\bar{c}_i(\mathbf{p}, \mathbf{u}) = 1 - \frac{c_i(\mathbf{p}, \mathbf{u})}{c_{\min}} \leq 0, \tag{3.6}$$

where $c_{\max}$ and $c_{\min}$ are the constraint limits.

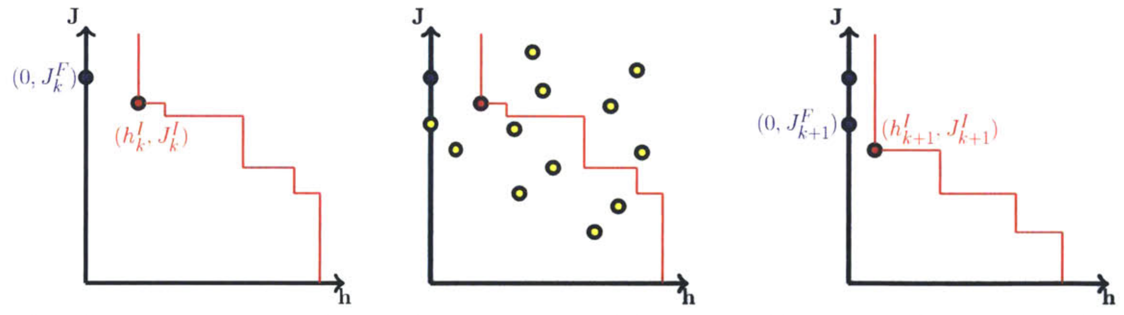
As can be readily seen, the optimal solution of $\chi(\mathbf{p}, \mathbf{u}; \wp)$ depends on the choice of the penalty parameter $\wp$. Thus ,the penalty parameter should be specified such that the constraint violation term has similar order of magnitude as the objective function. Poor choice of $\wp$ (either too large or too small) may lead to inaccurate results as the merit function in this case may not provide a good approximation of the objective function. In order to avoid poor choice of $\wp$, an iterative scheme is usually devised where $\wp$ is updated at every iteration until the solution of the merit function converges to that of the objective function. This iterative scheme might work well for certain cases where the cost of computing the objective function, and hence the merit function, is very low. However, it has limitations in realistic problems as it may incur prohibitive computational costs. Due to this limitation, we adopt a filter method as a way to handle nonlinear constraint, which we describe next.

Filter techniques are step acceptance techniques that provide the ability to overcome the pitfalls of penalty function methods. Such techniques treat the objective function and the constraint violation (infeasibility term) separately rather than combining them into a single function. Thus, the problem in (3.1) can be viewed as a bi-objective optimization where we seek to maximize/minimize the objective function $J(\mathbf{p}, \mathbf{u})$ and minimize the aggregated constraint violation $h(\mathbf{p}, \mathbf{u})$. Clearly, the second objective has a higher priority because every solution has to be feasible before being considered as optimal. Borrowing terminology from multi-objective optimization, a pair (J_a, h_a) is said to dominate another pair (J_b, h_b)—mathematically written as $(J_a, h_a) \prec (J_b, h_b)$—if and only if $J(\mathbf{p}, \mathbf{u}_a) \leq J(\mathbf{p}, \mathbf{u}_b)$ and $h(\mathbf{p}, \mathbf{u}_a) \leq h(\mathbf{p}, \mathbf{u}_b)$. The filter is defined as the list of pairs such that no pair dominates another pair. That is, an iterate $\mathbf{u}_k$ is acceptable to the filter if the pair (J_k, h_k) is not dominated by any other pair in the filter up to the k^{th} iteration. Detailed discussion and brief history of the filter method can be found in [41] and its application to constrained production optimization using MADS and PSO can be read in [58, 60, 62].

Before discussing the employment of filter methods in MCS, the definition of the word *'better'*, which has been mentioned extensively in previous section, should be revisited. For cases without nonlinear constraints, an egg is said to be better than another egg if it yields a higher objective function value. That is, an egg carrying the solution

$\mathbf{u}_m$ is better that an egg carrying the solution $\mathbf{u}_n$ if and only if $J(\mathbf{u}_m) \geq J(\mathbf{u}_n)$ for maximization problems. This definition is somewhat nested for cases with nonlinear constraints. That is, a better egg is an egg that carries a solution with a higher objective function value, in the case when both compared solutions are feasible, or an egg that carries a solution with a lower infeasibility term, in the case when both compared solutions are infeasible. If a feasible solution is compared to an infeasible solution, then the egg carrying the feasible solution is always better (a feasible solution is a solution that has a zero infeasibility term). It is clear here that the infeasibility term is always prioritized over the objective function value.

For the successful implementation of filter methods, a filter at iteration k constitutes the list of all non-dominated infeasible solutions in addition to the list of all feasible solutions up to the k^{th} iteration. At iteration k, two types of solutions are recorded in the filter: the best feasible solution $(0, J_k^F)$ and the least non-dominated infeasible solution (h_k^I, J_k^I). The filter is updated and the iteration is considered successful when new feasible solutions or when solutions that dominate some of the current filter solutions are found, as depicted in the sequence of illustrations in Figure 3.3.

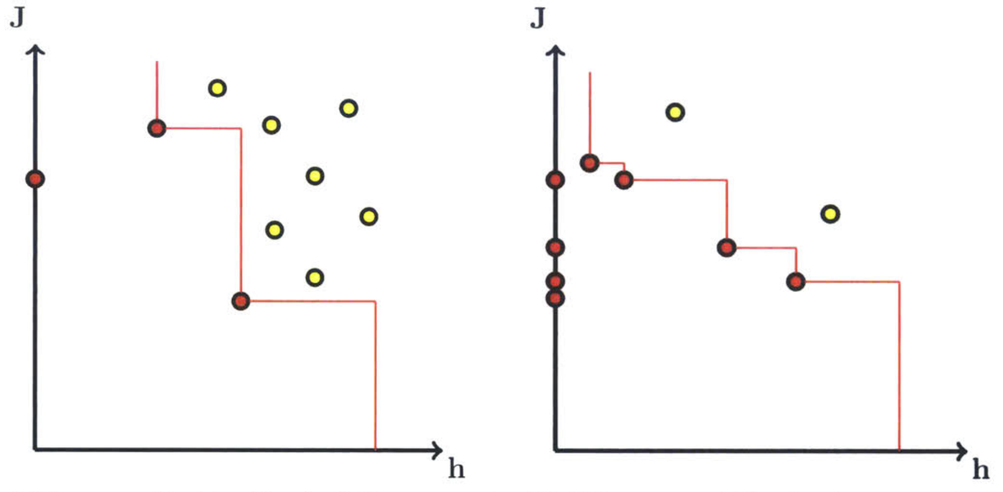


(a) Filter at the start of iteration k **(b) eggs evaluated during iteration k** **(c) Updated filter at iteration $k+1$**

Figure 3.3: Progress of filter form iteration k to $k+1$. The filter envelope moves toward the feasibility axis. (from Isebor [59] with modifications).

The modifications to the original MCS algorithm is mainly applied to how eggs are compared and categorized. In this work which considers nonlinear constraints, eggs are compared in terms of their feasibility first and then in terms of their closeness-to-feasibility (lower infeasibility). This condition favors the infeasibility term over the objective function value when two solutions are compared. Also, elite eggs are considered as eggs that are in the filter, i.e., feasible and non-dominated eggs, while non-elite eggs are those outside the filter. Algorithm 4 details these modifications.

The filter-based MCS provides a bias toward exploring the solution space at the beginning of the optimization run and gradually shifts that bias toward exploitation. During the first few generations, the number of elite eggs (feasible and non-dominated infeasible solutions inside the filter) is usually lower than the number of non-elite eggs, Figure 3.4(a). This translates into employing a higher volume of Lèvy flights to non-elite eggs which aids in exploring more promising areas of the solution space as opposed to employing crossover to elite eggs which provide local search. Toward the end of the optimization run, more eggs become elite, as shown in Figure 3.4(b), and accordingly a higher volume of crossover between elite eggs occurs. This provides a higher search resolution inside feasible regions and as a result improves solution quality.



(a) Elite eggs (inside filter) at the second generation (b) Elite eggs at the final generation

Figure 3.4: Illustration of the determination of elite and non-elite eggs as the optimization run progresses. Toward the end of the optimization run, the number of elite eggs inside the filter envelope increases promoting local search within the feasible search region (elite eggs are red, non-elite eggs are yellow).

Algorithm 4: Filter-based MCS

```
A ← Maximum Lèvy step size
$\varphi$ ← Golden ratio
$p_a$ ← fraction of eggs to be abandoned
Initialize a population of $n$ eggs $\mathbf{x}_i$ $(i = 1, 2, \cdots, n)$
for all $\mathbf{x}_i$ do
    calculate fitness $F_i = f(\mathbf{x}_i)$
end
Generation number $G \leftarrow 1$
while NumberofObjectiveEvaluations < MaxNumberofEvaluations do
    $G \leftarrow G + 1$
    Sort eggs by order of fitness
    Select the eggs to be discovered
    for all eggs outside filter do
        Current position $\mathbf{x}_i$
        Calculate Lèvy flight step size $\alpha \leftarrow A/\sqrt{G}$
        Perform Lèvy flight from $\mathbf{x}_i$ to generate new egg $\mathbf{x}_k$
        $\mathbf{x}_i \leftarrow \mathbf{x}_k$
        $F_i \leftarrow$ calculate fitness $f(\mathbf{x}_i)$
        $H_i \leftarrow$ calculate infeasibility $h(\mathbf{x}_i)$
    end
    for all eggs inside filter do
        Current position $\mathbf{x}_i$
        Pick another egg from the top eggs at random $\mathbf{x}_j$
        if $(\mathbf{x}_i = \mathbf{x}_j)$ then
            Calculate Lèvy flight step size $\alpha \leftarrow A/G^2$
            Perform Lèvy flight from $\mathbf{x}_i$ to generate new egg $\mathbf{x}_k$
            Calculate fitness $F_k = f(\mathbf{x}_k)$
            Calculate infeasibility $H_k = h(\mathbf{x}_k)$
            Choose a random nest $l$ from all nests
            if $(H_k \leq 0)$ and $(H_l \leq 0)$ then                // both solutions feasible
                if $(F_k < F_l)$ then
                    $\mathbf{x}_l \leftarrow \mathbf{x}_k$
                    $F_l \leftarrow F_k$
                    $H_l \leftarrow H_k$
                end
            else                                                     // at least one solution infeasible
                if $(H_k < H_l)$ then
                    $\mathbf{x}_l \leftarrow \mathbf{x}_k$
                    $F_l \leftarrow F_k$
                    $H_l \leftarrow H_k$
                end
            end
        else
            $dx = |\mathbf{x}_i - \mathbf{x}_j|/\varphi$
            Move distance $dx$ from the worst egg to the best egg to find $\mathbf{x}_k$
            Calculate fitness $F_k = f(\mathbf{x}_k)$
            Choose a random nest $l$ from all nests
            if $(H_k \leq 0)$ and $(H_l \leq 0)$ then                // both solutions feasible
                if $(F_k < F_l)$ then
                    $\mathbf{x}_l \leftarrow \mathbf{x}_k$
                    $F_l \leftarrow F_k$
                    $H_l \leftarrow H_k$
                end
            else                                                     // at least one solution infeasible
                if $(H_k < H_l)$ then
                    $\mathbf{x}_l \leftarrow \mathbf{x}_k$
                    $F_l \leftarrow F_k$
                    $H_l \leftarrow H_k$
                end
            end
        end
    end
    Abandon a fraction $p_a$ of the worst eggs and build new nests
end
```

3.3 Local Derivative-Free Approach for Well Placements

3.3.1 Mesh Adaptive Direct Search

Pattern search algorithms constitute a family of optimization algorithms that are derivative-free and can be used with black-box objective functions. They are generally based on the concept of polling, as depicted in Figure 3.5, which provides the ability to perform local exploratory moves on the objective function surface around current iterate. At any given iteration k, a poll stencil consists of the current best iterate $\mathbf{x}_k$ located at the center of the stencil and a set of trial points that are located at the stencil's extreme ends. The trial points form a pattern that is defined as:

$$M_k = \{\mathbf{x}_k + \Delta_k^m Dz : z \in \mathbb{N}^{nD}\}, \tag{3.7}$$

where $\mathbf{x}_k$ is the poll center, Δ_k^m is the underlying mesh size, and D is a matrix representing the set of positive spanning n_D directions $D \subset \mathbb{R}^n$.

The objective function is evaluated on all of the stencil end points—in addition to the already computed poll center. If any of the end points result in an improvement in terms of the objective function value over the poll center, then the center of the poll is shifted to the trial point that provides the highest improvement for the next iteration $k + 1$ and accordingly the stencil size is increased. If no trial point attains any improvement, the stencil size is reduced and polling continues with a smaller stencil size until an improvement is realized.

At each iteration, some positive spanning matrix $D_k \subset D$ dictates the orientation of the poll stencil. If this matrix is constant, i.e., $D_k = D$, the stencil orientation is fixed and the pattern search algorithm is called Generalized Pattern Search Method (GPS). If the stencil orientation varies from one iteration to the next, the algorithm is called Mesh Adaptive Search (MADS) [17]. A primary difference between MADS and GPS is the presence of an adaptive mesh underlying the polling stencil in MADS. The mesh is defined by its size Δ_k^m where $\Delta_k^m \leq \Delta_k^p$ and accordingly all stencil points must lie on the mesh (In GPS, there is a single stencil orientation with $\Delta_k^m = \Delta_k^p$). The mesh size typically decreases faster than the stencil size which allows the stencil to effectively access more possible directions.

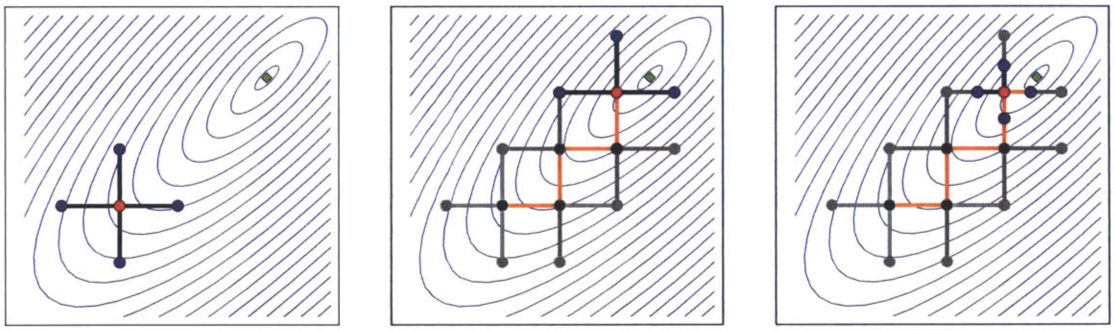

(a) Initial polling stencil (b) successive polling stencils (c) Final polling stencil showing contraction

Figure 3.5: Progress of polling in MADS algorithm in $\mathbb{R}^2$. The green diamond depicts the global optimum. The red circle is the poll center and the blue circles are the trial points. The orange staircase shows the history of the progress of the poll, (from Isebor [59] with modifications).

The advancement of the polling stencil for the MADS algorithm is depicted in Figure 3.6 for an optimization problem with two variables. Figure 3.6(a) shows the stencil's orientation at iteration k with different stencil and mesh sizes. If the polling process does not yield a successful iteration, i.e., none of the trial points lead to improvement over the poll center, then both the stencil size and the mesh size are reduced for the subsequent iteration $k + 1$ with a possible change of the stencil's orientation. The fact that Δ_k^p decreases faster than Δ_k^m provides the possibility for the stencil to reach out to more locations and thus search for more promising areas in the solution space.

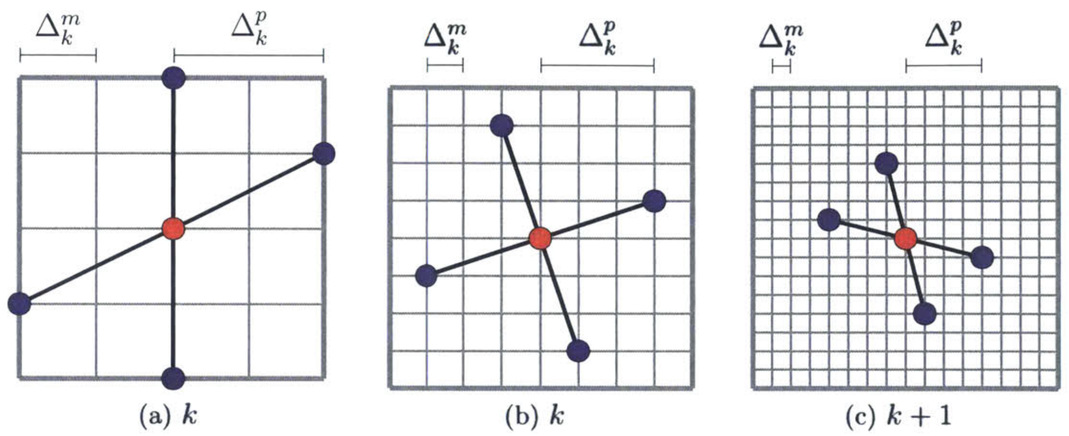


Figure 3.6: MADS stencil changing with iterations. Red circle represents the poll center while blue circles are the trial points. The polling stencil has the size Δ_k^p, and the mesh size is Δ_k^m, (from Isebor [59] with modifications).

Pattern search algorithms including MADS are local optimization algorithms which solution quality depends heavily on the initial solution guess. Therefore, MADS is often accompanied by an optional search step, in addition to the polling step, to provide global exploration of the solution space and lessen the localized behavior. The search step attempts to find a good starting point (poll center) for the MADS algorithm. Figure 3.7 illustrates the use of the search step $S_k = \{S_1, S_2, S_3, S_4\}$ to provide a good starting point for the polling step $P_k = \{P_1, P_2, P_3, P_4\}$. Algorithm 5 summarizes the basic MADS procedure.

In this work, we use contraction parameters $\eta = 0.75$ and $\psi = 0.5$ (determined via experimentation). This means the poll size is increased by a factor of 1.33 after successful iterations and decreased by a factor of 0.75 after unsuccessful iterations. The mesh size is doubled, up to the initial mesh size Δ_0 after successful iterations, and halved after unsuccessful iterations.

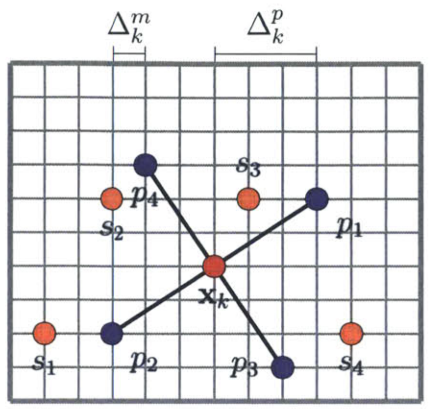


Figure 3.7: An example of polling step performed after a search step. The orange circles represent the iterates from the search step. The best search iterate is chosen as the poll center for MADS algorithm.

Two stopping criteria for the MADS algorithm are considered here. The MADS algorithm is terminated if Δ_k^p or Δ_k^m decrease beyond a specific threshold or if the maximum number of iterations is reached. In all of the examples presented in this work, the mesh size criteria terminates the MADS algorithm.

3.3.2 Constraint Treatment in MADS

To satisfy bound constraints, a projection operator, as in (3.2), is used to project solutions that fall outside the bounds of the problem. For General nonlinear constraints, we apply the filter method similar to that applied for the MCS algorithm. Filter methods have been studied in the context of MADS for solving nonlinear problems (NLPs) [16, 67]. This work is patterned after [16, 59] , with the addition of using ROM as a mean to evaluate trial points.

The filter technique is coupled with MADS algorithm through the determination of the poll center during the polling process. The poll center is considered as the highest feasible point in the filter, or the least infeasible point in the case that no feasible

Algorithm 5: MADS algorithm

```
Initializate :
Let $\mathbf{x}_0$ be the initial guess such that $f(\mathbf{x}_0)$ is finite and let $M_0$ be the mesh defined
over the solution space with initial mesh size $\Delta_0^p = \Delta_0^m = \Delta_0 > 0$ and let the
contraction parameters $0 < \psi < \eta < 1$.
while StoppingCriteriaNotMet do
1.  Search :
    Use a global search method, e.g., MCS, to find an improved solution such that
    $\mathbf{x}_{k+1} \in M_k$ defined by $\Delta_k^m$ and $f(\mathbf{x}_{k+1}) < f(\mathbf{x}_k)$
2.  Poll :
    If the search step was not successful, construct a poll set with a stencil size $\Delta_k^p$
    centered at $\mathbf{x}_k$, at random poll directions. Evaluate the objective function $f$ at
    the stencil's end points in order to find an improved solution $\mathbf{x}_{k+1}$
3.  Update :
    If either the search or poll step finds an improved solution, then let $\mathbf{x}_k = \mathbf{x}_{k+1}$
    and set $\Delta_{k+1}^p = \Delta_k^p/\eta$ and $\Delta_{k+1}^m = \Delta_k^m$ if $\Delta_k^p > \Delta_0$ else $\Delta_{k+1}^m = \Delta_k^m/\psi$.
    Otherwise, set $\mathbf{x}_{k+1} = \mathbf{x}_k$, $\Delta_{k+1}^p = \eta\Delta_k^p$ if $\Delta_k^p > \Delta_0$ else $\Delta_{k+1}^m = \psi\Delta_k^m$
end
```

point is available. A MADS iteration is considered successful if it yields an unfiltered point (a feasible or infeasible non-dominated point). In such a case, the stencil is extended and the filter is updated as new points are found that dominate some of the current points in the filter. Note that attaining a new unfiltered point does not necessarily indicate a change in the poll center as the new unfiltered point might not be the highest feasible or the least infeasible point. If the MADS iteration is unsuccessful, the mesh and poll size parameters are decreased and the filter remains the same.

As illustrated in [59, 9], the final filter can provide measurement of the sensitivity of the objective function to the constraint considered at no extra cost (similar analogy to the pareto frontier in multi-objective optimization). The sensitivity analysis can reveal information on how much a certain constraint has to be relaxed in order to achieve more gain in the objective function.

3.4 Hybrid MCS-MADS Approach Using ROM

Pattern search methods such as MADS are local methods in the sense that they achieve convergence (from arbitrary starting points) to points that fulfill necessary conditions for local optimality. It is therefore expected that MADS does not provide similar level of ability for global exploration that metaheuristic methods such as MCS usually provide—given a reasonable number of nests. In this work, we aim to capitalize on the global search nature of MCS and utilize MADS ability to rigorously converge to stationary points to create a hybrid search scheme that utilizes the positive aspects of both MCS and MADS to create a robust search approach. This approach is referred to as MCS-MADS hybrid approach.

The hybrid procedure presented in the work is an extension of the PSwarm algorithm [97], and the PSO-MADS hybrid algorithm [59]. The essential difference in the algorithm presented here is that we implement MCS instead of PSO due to the attractive features discussed in previous sections, and we use ROM instead of FOM during the polling step to lower the computational demands which is achieved by devising an efficient systematic approach for ROM construction and verification. Figure 3.8 illustrates the general workflow of the MCS-MADS hybrid procedure.

The hybrid procedure specifies the number of nests, including any user-defined initial guess, and then applies one iteration of MCS. The role of MCS is to provide a robust search strategy that outperforms the one originally depicted during the MADS search step. Consecutive iterations where the search step is deemed successful are equivalent to consecutive iterations of the standalone MCS. A MCS search step is deemed successful if a new best egg is found. That is, a new best solution that dominates the previous best solution inside the filter in terms of objective function J and constraint violation h. This definition of success is different from that used in MADS, where any iterate that leads to an unfiltered point (update in the filter) is considered successful. This stringent criterion is mainly applied to avoid performing many expensive MCS iterations— note that iterates in the search step are evaluated using FOM. One main filter is used in this approach—accessible to both MCS and MADS—as opposed to PSO-MADS algorithm where multiple subfilters are used. The filter in MCS serves as a mean to classify the type of eggs (elite or non-elite)—in addition to nonlinear constraint treatment.

When the search step is designated unsuccessful, MADS poll step is performed. The poll is centered around the best egg achieved by MCS in the search step. Projection bases are then constructed using the poll center iterate as a root-point to build the ROM. Each subsequent poll trial iterate is accordingly evaluated using the ROM instead of FOM. ROM is chosen as the evaluation mean in the poll step as the distance between the training root-point (poll center) and the poll trial iterates is controlled by the poll size and therefore we can ensure controlled deviations from the root-point—hence more accurate ROM results. Polling continues as long as consecutive poll steps are successful. Success criteria here is dictated by the possibility of finding an unfiltered solution (better feasible or non-dominated points) and the ability of ROM to accurately predicts the descent direction of the algorithm (theoretically, both FOM and ROM should predict descent direction). The best poll iterate—the iterate that yields a better solution than the poll center—is then evaluated using FOM. This additional run is made to evaluate the ROM's ability to predict the algorithm's attempt to find a descent direction using ROM, and to prepare for the next stencil as this best iterate will be the poll center for the next poll step. ROM performance is evaluated using:

$$\kappa = \frac{J_k(\mathbf{x}_k + \Delta_k^m Dz) - J_k(\mathbf{x}_k)}{J_k^{\mathcal{R}}(\mathbf{x}_k + \Delta_k^m Dz) - J_k^{\mathcal{R}}(\mathbf{x}_k)}, \tag{3.8}$$

where $J_k(\mathbf{x}_k + \Delta_k^m Dz)$ is the new best iterate evaluated using FOM, $J_k(\mathbf{x}_k)$ is poll center iterate evaluated using FOM, $J_k^{\mathcal{R}}(\mathbf{x}_k + \Delta_k^m Dz)$ is the new best iterate evaluated using ROM, and $J_k^{\mathcal{R}}(\mathbf{x}_k)$ is the poll center evaluated using ROM. The ROM performance parameter κ determines how well the ROM follows the descent direction. In a sense, if the performance parameter is larger than a certain threshold ϵ, it indicates that the ROM approximates the FOM very well and thus the poll step is deemed successful (note the presence of a nested success criteria; first succeeding in finding a new unfiltered iterate, then determining if the direction given by ROM using this iterate coincides with the direction given by the FOM). If the performance parameter is less than ϵ, it indicates that the ROM either fails in providing good approximation to the FOM ($0 < \kappa < \epsilon$) or fails to predict the descent objective function direction ($\kappa < 0$). Hence, the new unfiltered iterate is not actually an unfiltered iterate as the current ROM is not obviously a competent evaluation model. In this case, the poll step is considered unsuccessful and the poll and mesh sizes are reduced and the hybrid algorithm returns to MCS with the current poll center iterate replacing the best egg in the previous generation. It is important to note here that the updated MCS best egg is already evaluated using FOM (consistent with all other eggs in the current generation) to avoid any discrepancy—ROM after all provides approximate results that are slightly higher or lower than the actual FOM results. MCS search step is terminated when the number of MCS generations is reached and the overall hybrid algorithm is terminated when the mesh size decreases below a prescribed tolerance Δ_{tol} or when the maximum number of hybrid evaluations k_{max} is reached.

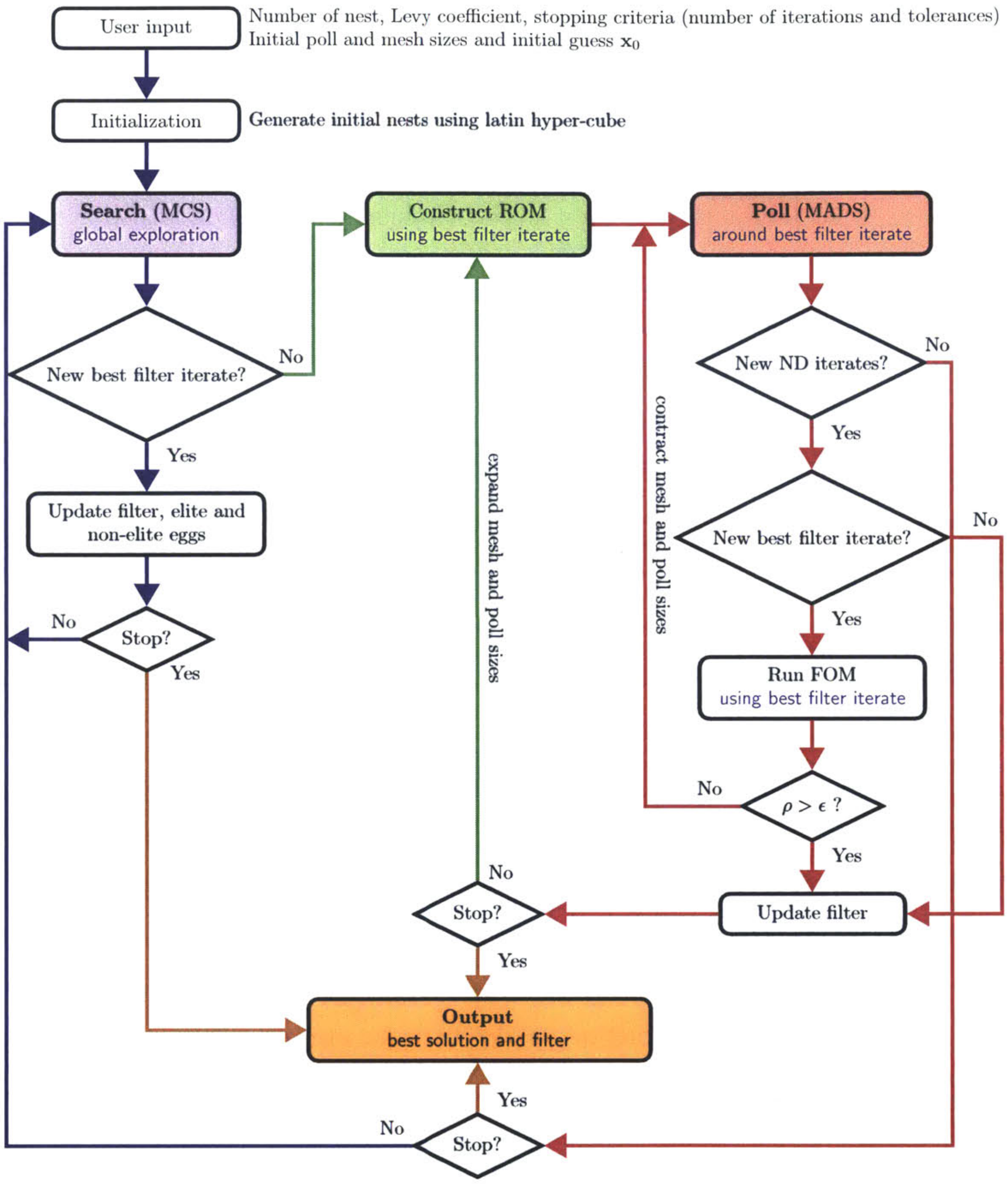

User input
Number of nest, Levy coefficient, stopping criteria (number of iterations and tolerances)
Initial poll and mesh sizes and initial guess $\mathbf{x}_0$
Initialization
Generate initial nests using latin hyper-cube
Search (MCS)
global exploration
Construct ROM
using best filter iterate
Poll (MADS)
around best filter iterate
New best filter iterate?
No
Yes
Update filter, elite and non-elite eggs
Stop?
No
Yes
expand mesh and poll sizes
contract mesh and poll sizes
New ND iterates?
No
Yes
New best filter iterate?
No
Yes
Run FOM
using best filter iterate
$\rho > \epsilon$?
No
Yes
Update filter
Stop?
No
Yes
Output
best solution and filter
Yes
No
Stop?

Figure 3.8: Flowchart of MCS-MADS hybrid implementation using ROM for local search (blue arrows depict optimization within MCS, red arrows indicate optimization within MADS, green arrows represent the use of ROM as the search engine, and orange arrows indicate termination of the run).

3.5 Derivative Approach for the Well Controls

A target application of reduced-order modeling is to be able to make a rational informed decision in a reasonable time. Within the context of reservoir management, this includes the ability to apply changes in production strategy to maximize recovery factor or NPV for instance. ROMs are suitable replacement to FOMs and are sufficiently accurate only in a restricted zone around the point in decision variable space, e.g., BHP well controls employed to construct the ROM, where they are constructed. Consequently, the ROM needs to be updated in a systematic manner over the course of the optimization and decision making process. In this section, we describe an adaptive framework using ROM based on trust-region methodology that allows for an automatic update and validation of the ROM during the optimization process.

3.5.1 ROM-based Trust-region Framework

Trust-region algorithm is a class of relatively new algorithms that depends on gradient information (objective function gradients with respect to the controls) to determine optimal solutions. The objective function gradients used in this work are computed using an adjoint procedure. Refer to [88] for detailed description of adjoint procedures in reservoir simulation. Unlike line search methods where a step direction is first chosen then a step size is determined, the trust-region method first choses a step size, referred to as the size of the trust-region, then determines a step direction to achieve optimal solutions.

The trust-region approach is strongly associated with approximate models. These approximate models are *'trusted'* only within the vicinity of a region surrounding an iterate $\mathbf{u}_k$. This is reasonable since approximate models for nonlinear functions, e.g., quadratic models or reduced-order models, can accurately approximate the original models locally. It is for this reason that trust-region approach provides the natural choice for ROM-based optimization.

The Trust-region method is considered an excellent adaptive framework for ROM-based optimization. It does not only provide the mean to restrict the optimization step within the ROM's validity, but also synchronizes ROM updates with information obtained during the course of the optimization, thus providing a robust and globally convergent framework. To see this, let's start at the k^{th} iteration of an optimization cycle. ROM is constructed at a particular training point, $x_k = \{p_k, S_k\}$, obtained using the control $\mathbf{u}_k$ which is used to build the model function for the trust-region subproblem. We define the ROM-based trust-region subproblem at iteration k as:

$$\begin{aligned} \max_{\boldsymbol{\delta}\in\mathbb{R}^n} \quad & J_k^{\mathcal{R}}(\mathbf{u}_k+\boldsymbol{\delta}), \\ \text{s.t.} \quad & \mathbf{F}_k^{\mathcal{R}}(\hat{\mathbf{x}}, \mathbf{u}_k+\boldsymbol{\delta}) = 0 \\ & \mathbf{g}_k^{\mathcal{R}}(\mathbf{u}_k+\boldsymbol{\delta}) = 0 \\ & \mathbf{h}_k^{\mathcal{R}}(\mathbf{u}_k+\boldsymbol{\delta}) \geq 0 \\ & \mathbf{L}_{\mathrm{B}} \leq \mathbf{u}_k+\boldsymbol{\delta} \leq \mathbf{U}_{\mathrm{B}} \\ & \| \boldsymbol{\delta} \|_\infty \leq \triangle_k \end{aligned} \tag{3.9}$$

where $J_k^{\mathcal{R}}(\mathbf{u}_k+\boldsymbol{\delta})$ is the reduced objective function we wish to maximize computed using ROM, $\mathbf{F}_k^{\mathcal{R}}(\hat{\mathbf{x}}, \mathbf{u}_k+\boldsymbol{\delta})$ is the set of adjoint constraints, $\mathbf{g}_k^{\mathcal{R}}(\mathbf{u}_k+\boldsymbol{\delta})$ and $\mathbf{h}_k^{\mathcal{R}}(\mathbf{u}_k+\boldsymbol{\delta})$ are the equality and inequality constraints respectively computed from the reduced set of state variables of the ROM. $\mathbf{L}_{\mathrm{B}}$ and $\mathbf{U}_{\mathrm{B}}$ are upper and lower control boundaries and $\triangle_k$ is the control trust-region at iteration k. Finally, $\boldsymbol{\delta}$ is the trial step representing the increment in controls variable we want to optimize.

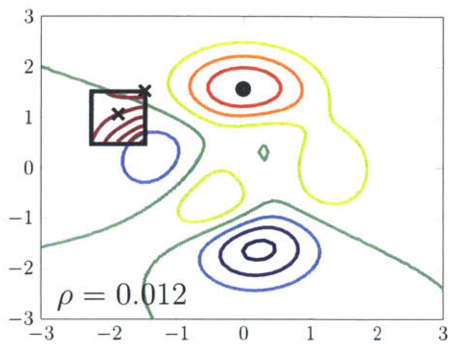
ROM-based trust-region algorithm starts by constructing a ROM employing procedures described in Chapter 2 while using an initial control $\mathbf{u_0}$. The basis functions are computed using snapshots obtained from the FOM. Then (3.9) is used to compute the optimal $\boldsymbol{\delta}$ that maximizes a quadratic reduced objective function where $\boldsymbol{\delta}$ is bounded by a trust region $\triangle_k$ that is initially defined by the user. The trust-region is centered at the current iterate, i.e., $\mathbf{u_0}$, and initialized in such a way that the ROM's results are very close to that obtained using the FOM for any iterate inside the trust-region.

The key content of a trust-region algorithm is how to deem a new iterate successful. For an unconstrained problem, two conditions must hold to declare a new iterate successful. The first condition dictates that a new iterate, $\mathbf{u}_{k+1} = \mathbf{u}_k + \boldsymbol{\delta}$, is considered successful if it provides sufficient improvement in the objective function value, i.e., $J_{k+1} < J_k$ for minimization case. The second condition relates to the accuracy of the ROM used to find the trial step. Since the new iterate $\mathbf{u}_{k+1}$ is different from the ROM root-point—iterate $\mathbf{u}_k$ used to construct the ROM, it is important that the ROM yields similar magnitude of reduction in the objective function as that realized using the FOM. This is examined via a calibration parameter:

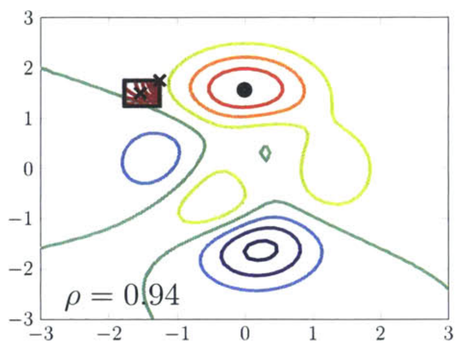
$$\rho_k = \frac{\text{ared}_k}{\text{pred}_k} = \frac{J_k(\mathbf{u_k} + \boldsymbol{\delta}) - J_k(\mathbf{u_k})}{J_k^{\mathcal{R}}(\mathbf{u_k} + \boldsymbol{\delta}) - J_k^{\mathcal{R}}(\mathbf{u_k})} \leq \eta_1, \tag{3.10}$$

where ared_k is the actual reduction using FOM and pred_k is the predicted reduction using ROM. In addition to determining the success of a new iterate (represented by improvement in objective function), the calibration parameter dictates how well the ROM approximates the FOM and whether any retraining is necessary for subsequent trust-region steps. That is, a large ρ_k indicates that the ROM approximates the FOM very well and thus the trust-region should be enlarged for subsequent steps. Otherwise when ρ_k is small, the new iterate may get rejected and the trust-region should be reduced as it is an indication that the ROM poorly approximates the

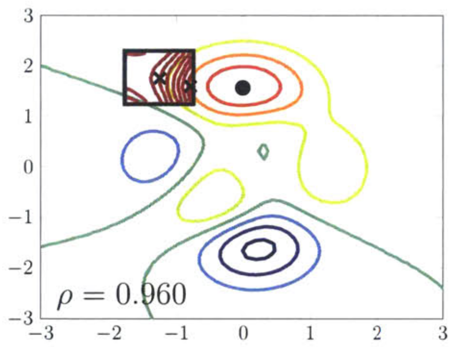
FOM. This process is depicted by the sequence of illustrations in Figure 3.9 for an optimization problem with two variables. The size of the trust-region adaptively changes as a function of the calibration parameter (according to how well the ROM can approximate the FOM). Each trust-region is centered around the initial iterate for the subproblem.



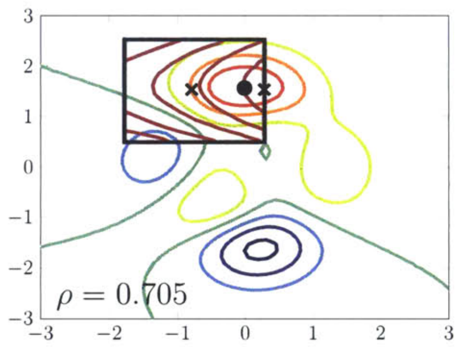
(a) Initial trust-region.



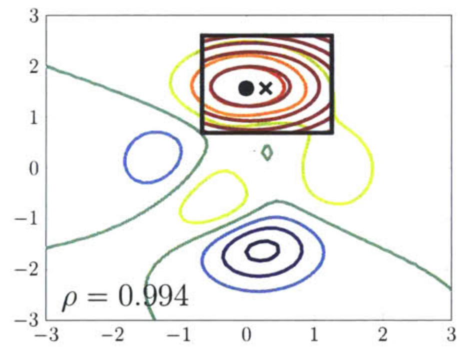
(b) Contraction of trust-region due to poor ROM approximation.



(c) Expansion of trust-region due to excellent ROM approximation.



(d) Trust-region reaches maximum allowable size $\triangle_{\max}$.



(e) Trust-region subproblem solution converge to the optimal solution.

Figure 3.9: Progress of trust-region algorithm for an optimization problem in $\mathbb{R}^2$. The contours inside the box trust-region are the contours of the objective function approximated using ROM while the contours outside the trust-region are those computed using FOM (from Willcox [105] with modifications).

3.5.2 Constraint Treatment in Trust-region Algorithm

To satisfy bound constraints, a projection operator, as in (3.2), is used to project solutions that fall outside the bounds of the problem. For general nonlinear constraints, gradient-based optimization algorithms typically apply approaches that transform the problem into an unconstrained optimization problem with a penalty function formulation that penalizes infeasible solutions. These approaches include penalty functions, barrier functions, augmented Lagrangian functions, and quadratic programming methods [36]. In our work, we use a different approach, namely the filter technique, to treat nonlinear constraints during the trust-region optimization subproblem.

Filter methods have been studied in the context of line-search methods for solving nonlinear problems (NLPs) [87, 41, 100, 40, 101] and are incorporated into the NLP solver IPOPT [1]. This work is patterned after Agarwal and Biegler [4] trust region filter method using ROM, with additional modifications for DEIM-based reduced-order modeling.

For problems without nonlinear constraints, a successful iterate is determined first based on the improvement in the objective function value, and then based on the true improvement dictated by comparing the objective function value of the ROM used in the trust-region subproblem to that of the FOM. For problems with nonlinear constraints; however, we only accept a new iterate which corresponding pair (J_{k+1}, h_{k+1}) is sufficiently outside the current filter, and hence sets an envelope around the filter (shown by the dashed line in Figure 3.10). As in [101], If the new iterate is accepted, then we set $\mathbf{u}_{k+1} = \mathbf{u}_k + \boldsymbol{\delta}$, and possibly increase the size of the trust-region and update the filter, i.e., the following condition holds:

$$h(\mathbf{u}_k + \boldsymbol{\delta}) \leq (1-\gamma_h)h_l \quad \text{or} \quad J(\mathbf{u}_k + \boldsymbol{\delta}) \leq J_l - \gamma_J h_l \quad \forall (h_l, J_l) \in \mathcal{F}_k, \qquad (3.11)$$

where $\gamma_h, \gamma_J \in (0,1)$ are chosen to be small, and $\mathcal{F}_k$ is the filter up to the k^{th} iteration. In contrast, if the new iterate is dominated by the current filter, then we reject it, set $\mathbf{u}_{k+1} = \mathbf{u}_k$, and resolve the trust-region subproblem with a reduced trust-region this time.

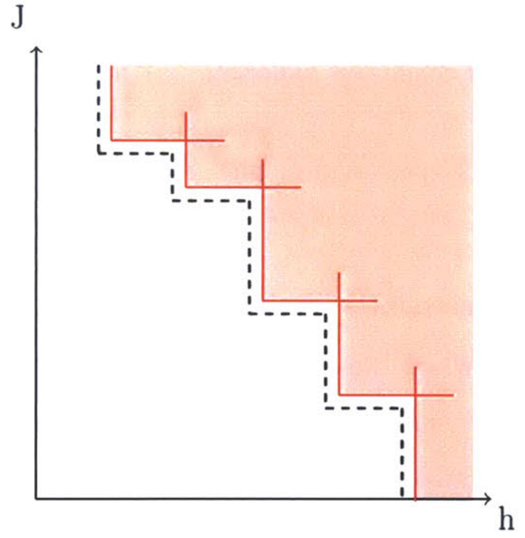
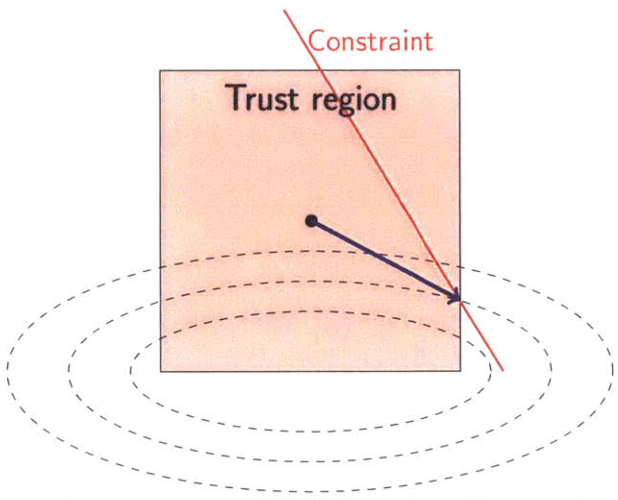
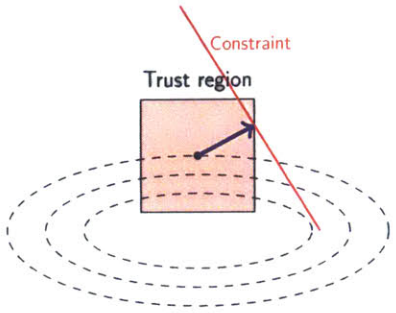


Figure 3.10: Filter for trust-region algorithm. All iterates that are below and to the left of the envelop (dashed line) are acceptable to the filter.

By reducing the trust-region, Problem (3.9) may become inconsistent (objective function and constraint violation act in opposite directions) as illustrated by halving the trust-region in Figure 3.11. This inconsistency is regarded as an indication that the current iterate is too far from the feasible region to make meaningful progress toward an optimal solution. Hence, we concentrate in this subproblem on minimizing the constraint violation $h(\mathbf{u}_k)$ in an attempt to reach feasibility. This step is referred to feasibility restoration step [41].



(a) Trust-region subproblem minimizing both objective function and constraint violation.



(b) Trust-region subproblem minimizing constraint violation only.

Figure 3.11: Illustration of trust-region algorithm switch to feasibility restoration step.

As observed in [4], the constraint violation $h(\mathbf{u}_k)$ in condition (3.11) dominates to a certain point especially when infeasibility is large. However, this condition alone does not necessarily ensure convergence to a stationary point, i.e., an iterate might converge to an arbitrary feasible solution. Therefore, upon entering the feasible region ($h(\mathbf{u}_k) = 0$), we focus on descent of $J(\mathbf{u}_k)$, i.e., switch to the following condition:

$$J_k(\mathbf{u_k} + \boldsymbol{\delta}) - J_k(\mathbf{u_k}) \leq 0, \tag{3.12}$$

and the algorithm reduces to a classical unconstrained trust-region approach. For this case, no filter point is added. Details of the procedure is outlined in Algorithm 6.

Algorithm 6: ROM-Based Trust-region Algorithm

```
(a) Initialization:
Chooese $0 < \eta_1 < \eta_2 < 1 \leq \eta_3$, $0 < \gamma_1 \leq \gamma_2 < 1 \leq \gamma_3, 0 < \gamma_h, \gamma_J \leq 1$
Specify initial trust-region radius $\triangle_0$, minimum radius $\triangle_{\min}$, and maximum radius $\triangle_{\max}$.
Compute state snapshot matrix $\mathbb{X}_0$, based on initial control $\mathbf{u}_0$ and compute $J_0(\mathbf{u_0})$
Set $k = 0$
(b) Construct ROM:
Compute basis based on snapshot $\mathbb{X}_k$ and construct ROM
(c) Step Calculation:
Compute step $\delta_k$ by solving trust-region subploblem (3.9):
    • If the new iterate is unfiltered, i.e., (3.11) is satisfied:
      proceed to (d).
    • Else:
      set $\mathbf{u}_{k+1} = \mathbf{u}_k$,
      update trust-region radius $\triangle_{k+1} = \gamma_1 \triangle_k$,
      set $k = k + 1$ and go to (c).
(d) Compute Calibration ratio:
Compute new snapshot $\mathbb{X}_+$ based on $\mathbf{u}_k + \delta_k$, and evaluate full objective function $J_k(\mathbf{u_k} + \delta_\mathbf{k})$
Define calibration ratio as:
    $$\rho_k = \frac{\text{ared}_k}{\text{pred}_k} = \frac{J_k(\mathbf{u}_k + \delta_k) - J_k(\mathbf{u}_k)}{J_k^{\mathcal{R}}(\mathbf{u}_k + \delta_k) - J_k^{\mathcal{R}}(\mathbf{u}_k)}$$
(e) Trust-region Evaluation and Update:
    • If $\rho_k \geq \eta_3$:
      set $\mathbf{u}_{k+1} = \mathbf{u}_k + \delta_k$,
      $J_k(\mathbf{u_{k+1}}) = J_k(\mathbf{u_k} + \delta_\mathbf{k})$,
      $\mathbb{X}_{k+1} = \mathbb{X}_+$,
      update trust-region radius $\triangle_{k+1} = \min(\triangle_{\max}, \gamma_2 \triangle_k)$,
      set $k = k + 1$ and go to (b).
    • If $\eta_2 \leq \rho_k < \eta_3$:
      set $\mathbf{u}_{k+1} = \mathbf{u}_k + \delta_k$,
      $J_k(\mathbf{u_{k+1}}) = J_k(\mathbf{u_k} + \delta_\mathbf{k})$,
      $\mathbb{X}_{k+1} = \mathbb{X}_+$,
      update trust-region radius $\triangle_{k+1} = \triangle_k$,
      set $k = k + 1$ and go to (b).
    • If $\eta_1 \leq \rho_k < \eta_2$:
      set $\mathbf{u}_{k+1} = \mathbf{u}_k + \delta_k$,
      $J_k(\mathbf{u_{k+1}}) = J_k(\mathbf{u_k} + \delta_\mathbf{k})$,
      $\mathbb{X}_{k+1} = \mathbb{X}_+$,
      update trust-region radius $\triangle_{k+1} = \gamma_2 \triangle_k$,
      set $k = k + 1$ and go to (b).
    • If $\rho_k < \eta_1$:
      set $\mathbf{u}_{k+1} = \mathbf{u}_k$,
      update trust-region radius $\triangle_{k+1} = \gamma_1 \triangle_k$,
      set $k = k + 1$ and go to (b).
```

Chapter 4

Management and Optimization of Realistic Fields Using Reduced-Order Models

This chapter extends the application of reduced-order models to two realistic and complex three-dimensional reservoir models. The models contain $O(10^5)$ grid blocks and represent sections from different parts of a giant carbonate field located in the Middle East.

Field development optimization studies are carried out using both models. For each study, the optimization is performed sequentially. That is, optimal well locations are first determined with a specific set of production controls. Then, the optimal well locations are used to determine the optimal set of well controls. Reduced-order modeling techniques are used to construct reduced versions of the full models and accelerate/drive the optimization process. The approach still requires running the FOM during the optimization study in order to update the ROM and ensure accurate results.

This chapter demonstrates the applicability of ROMs for reservoir management decision-making on complex reservoirs. Specifically, as shown later in this chapter, ROMs can provide vast reductions in computational costs while achieving closely comparable results to those obtained using FOMs. As a result, computational savings can be of multiple order of magnitudes allowing for opportunities to examine extensive range of field development strategies.

Also, we compare the performance of the MCS, and MCS-MADS approaches coupled with ROM and highlight the effectiveness of the filter method for nonlinear constraint handling. We show that using the hybrid MCS-MADS accelerated by ROMs can improve the optimization outcome with only a minor increase in computational cost.

4.1 Pierce Model: Heterogeneous Carbonate Reservoir

4.1.1 Description of the Simulation Model

The simulation model, referred to as Pierce model, comprises a sector model of an onshore field located in the Middle East. The field is a giant anticline trap that produces from two main reservoirs. These reservoirs are upper and lower carbonate reservoirs separated by a thick non-reservoir formation (shale). The upper reservoir, considered in this example, is prolific throughout the entire field. The model contains a total of 25,194 grid blocks ($N_x = 51, N_y = 19, N_z = 26$). The physical dimension of each grid block is 250 m $\times$ 250 m $\times$ 250 m. Figure 4.1 and 4.2 depicts the grid structure, initial pressure and saturation. The model has aquifer support along the east and west flanks while other boundaries are modeled as no flow. There are nine production wells (designated P_1–P_9) located at the crest of the model and four injection well (designated I_1–I_4) drilled in the flanks. Production wells are completed in the top two layers while injection wells are completed at the most bottom three layers. All of the wells are under bottom hole pressure control.

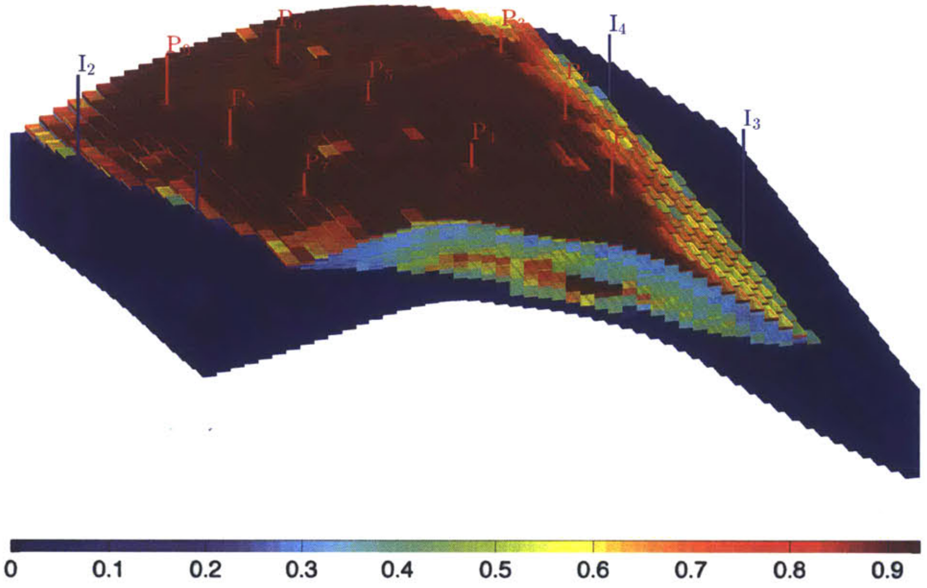


Figure 4.1: Three dimensional grid structure of Pierce model with four injectors placed initially at the flanks and nine producers placed at the crest of the field. Initial oil saturation is shown in the background. Significant amount of oil still exists in the top layers.

Figure 4.3 shows the permeability in the x-direction for selected layers (layer number is indicated above each figure). The permeability is taken to be a diagonal tensor, with $k_x = k_y$. The mean k_x and k_z are 546 mD and 11.6 mD respectively while the mean porosity is 0.16. For oil, we set $\rho_o = 45$ lb/ft^3, $\mu_o = 3$ cp; for water, we set $\rho_w = 65$ lb/ft^3, $\mu_w = 1$ cp. The system is incompressible and capillary pressure is neglected in this example. The relative permeabilities for the oil and water phases are specified using (2.47) and (2.48) with $k^0_{ro} = k^0_{rw} = 1$ and $a = b = 2$.

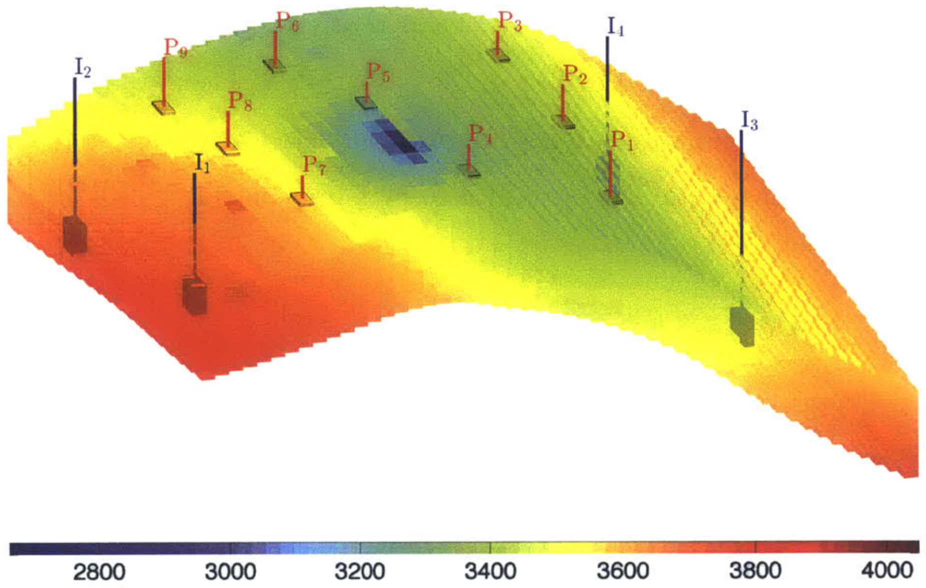


Figure 4.2: Three dimensional grid structure of Pierce model with four injectors placed initially at the flanks and nine producers placed at the crest of the field. Initial pressure (in psi) is shown in the background at datum depth of 6500 ft.

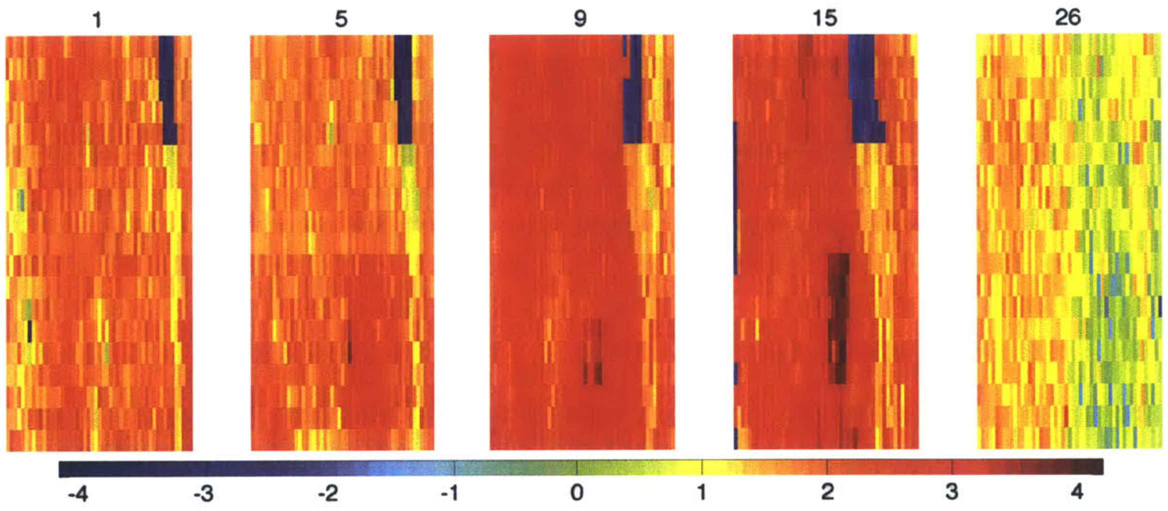


Figure 4.3: Permeability in the x-direction ($\log_{10} k$ with k expressed in mD) for selected layers in Pierce model.

4.1.2 Training and Testing Results Using ROM

The FOM is used to perform a training simulation run in order to collect state solutions (pressure and saturation) and nonlinear term (flux) snapshots necessary to construct the ROM. During the training simulation, all injection and production wells operate at a constant bottom hole pressure of 8,000 and 1,000 psi respectively. The training case is simulated for 5000 days using MRST, and a total of 600 snapshots are collected and used to build the orthogonal bases.

The basis matrices are constructed following the POD approach described in (2.2) for pressure, saturation, and flux. The reduced pressure, saturation, and flux bases contain 55, 130, and 110 columns respectively. Thus, we reduce the dimension of the problem from $50,388$ unknowns (considering pressure and water saturation as solution states for each grid block) to only 185 unknowns. Note that determining the number of basis remains to be a heuristic procedure as the omitted energy criteria, described in (2.2), still does not guarantee accuracy nor stability. In order to retain nonlinearity at a reduced dimension, the flux basis matrix is used to determine the location of interpolation points necessary to reconstruct the flux field. A total of 660 interpolation points are selected using Algorithm 3 in addition to the corresponding neighboring grid block (relational links) necessary for upstream weighting. The spatial distribution of interpolation points is illustrated in Figure 4.4. Mostly, interpolation points are situated in areas of high permeability (area bounded by $\mathrm{P}_5, \mathrm{P}_6, \mathrm{P}_8$, and P_9). This is where the highest change in flux occurs which is clearly supported by the high production rate of wells located within the vicinity of that area (see Figure 4.6).

Figure 4.5 indicates the bottom hole pressure test schedule used to assess the accuracy of the ROM. The bottom hole pressure control for injection wells varies between 4,000 psi and 10,000 psi while it varies between 500 psi and 2,500 psi for production wells.

The bottom hole pressure control is perturbed every 625 days comprising a total of eight perturbations.

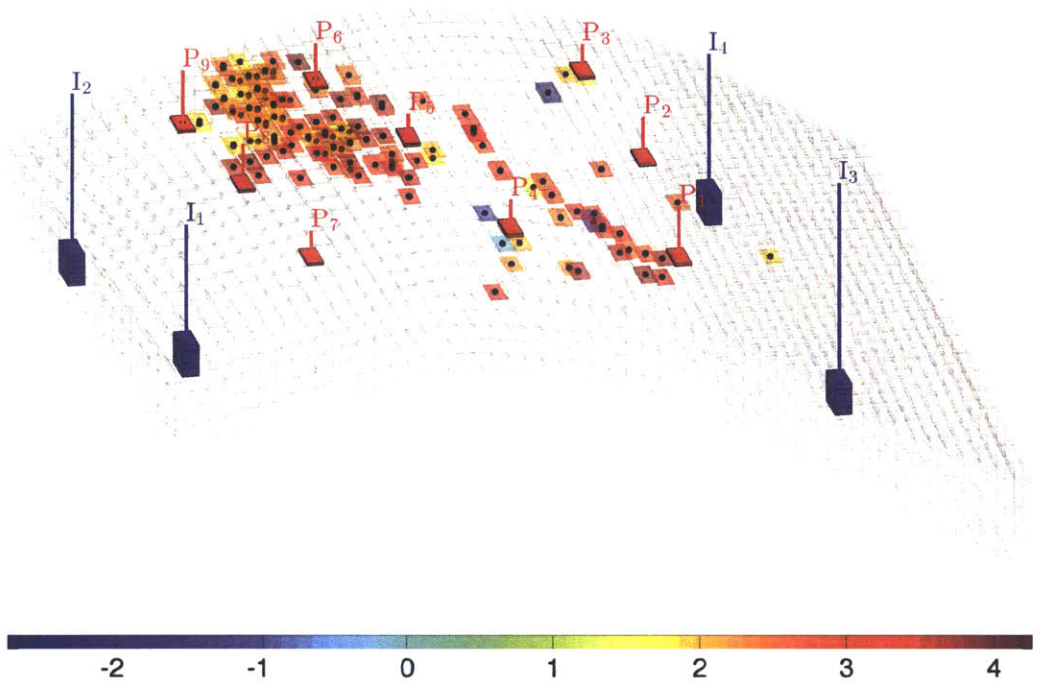
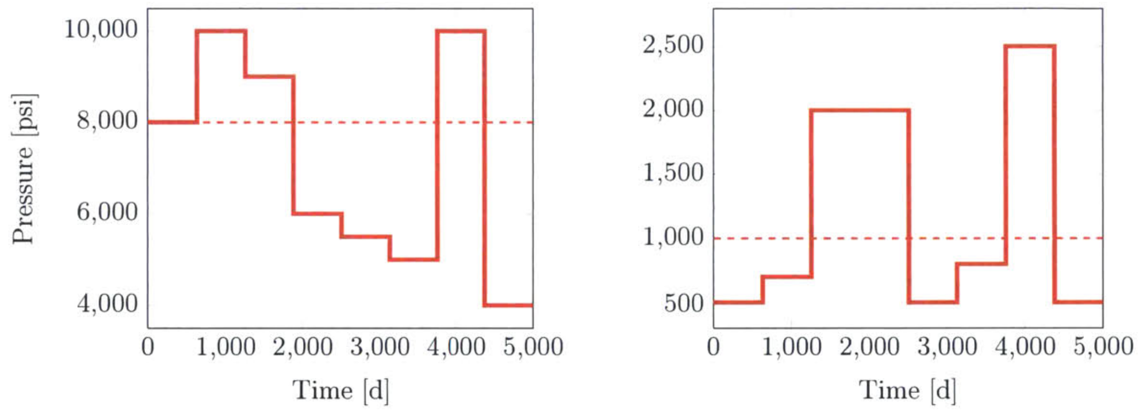


Figure 4.4: Spatial distribution of interpolation points selected to reconstruct flux field for Pierce model (background displays $\log_{10} k_x$ for grid blocks determined by DEIM Algorithm). Most DEIM points are situated in areas of high permeability (area bounded by P_5, P_6, P_8, and P_9)

The ROM performance is now examined using the testing production and injection schedules. The ROM is constructed using the three basis matrices generated previously in addition to the interpolation points which are used to keep nonlinearity computation cost at minimum. Figure 4.6 shows oil and water production rates from the nine production wells while Figure 4.7 displays water injection from the four injection wells. In this and subsequent examples, the dashed lines show the training simulation solution, which controls were used to build the ROM, the solid lines display the reference test case solution simulated using the FOM, and the circles depicts the test case solution simulated using the ROM. Clearly, the nine production wells contribute to a wide range of production trajectories ranging from excessive water production, e.g., P_1, to minimal water production, e.g., P_5, in addition to high and low overall fluid rate, e.g., $O(100) - O(1000)$. The ROM generally demonstrates close agreement with the reference full-order simulation although the computational costs are substantially different. The simulation runtime for the full-order model is about 1550 seconds while the ROM, in contrast, takes about 13 seconds. This results in a speed up factor of about 119.



(a) Injection BHP schedule (b) Production BHP schedule

Figure 4.5: Injection and production schedules for training and testing cases for the Pierce model. Dashed lines represent training BHP controls while solid lines depict testing BHP controls. The training schedule is chosen to be constant while the training schedule is perturbed every 625 days comprising a total of eight perturbations.

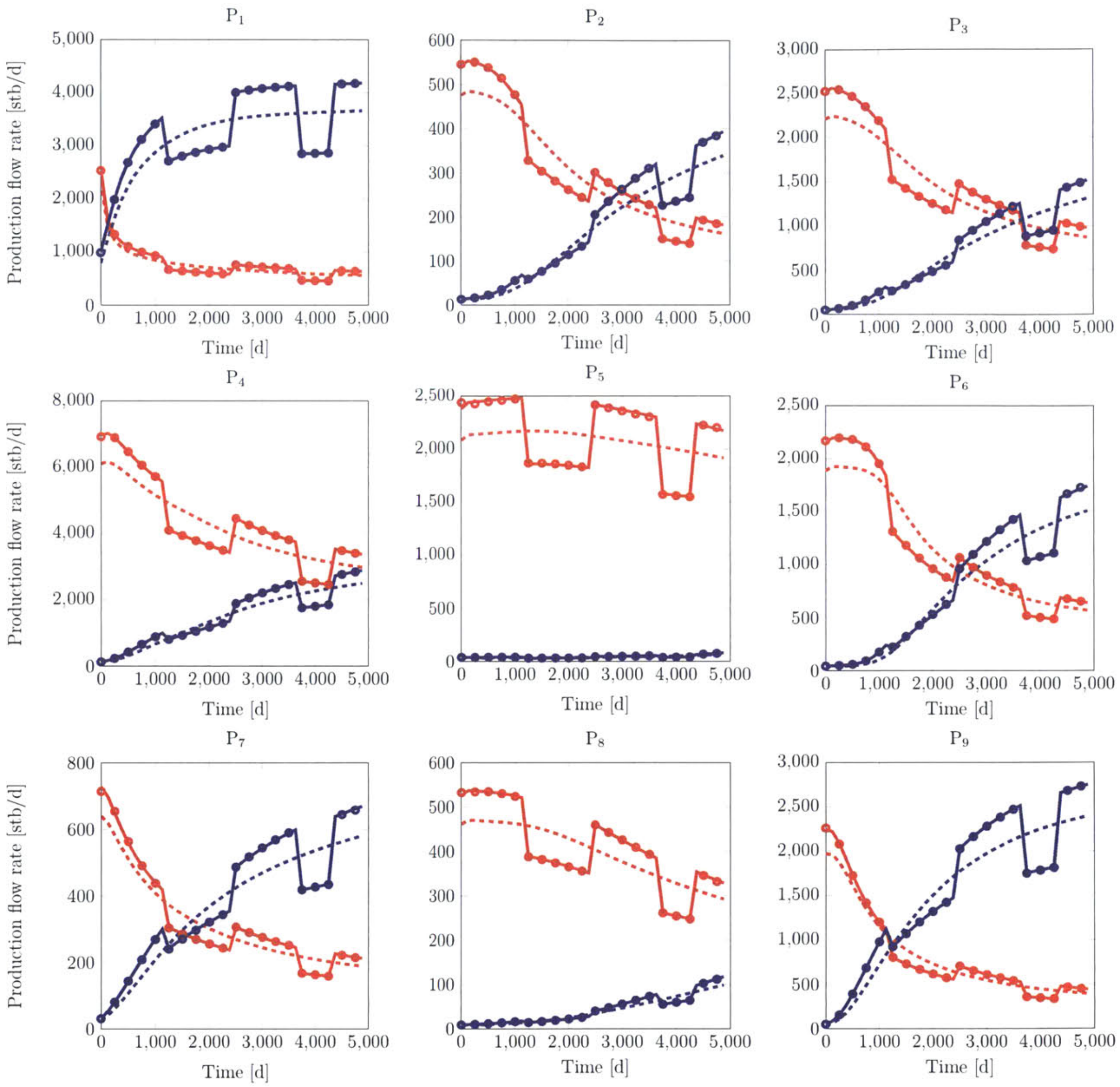


Figure 4.6: Production rates for each well in Pierce model. Dashed line represents training run trajectory, circles indicate ROM results using testing schedule, and solid line represents the reference testing run using FOM. Red designates oil while blue designates water.

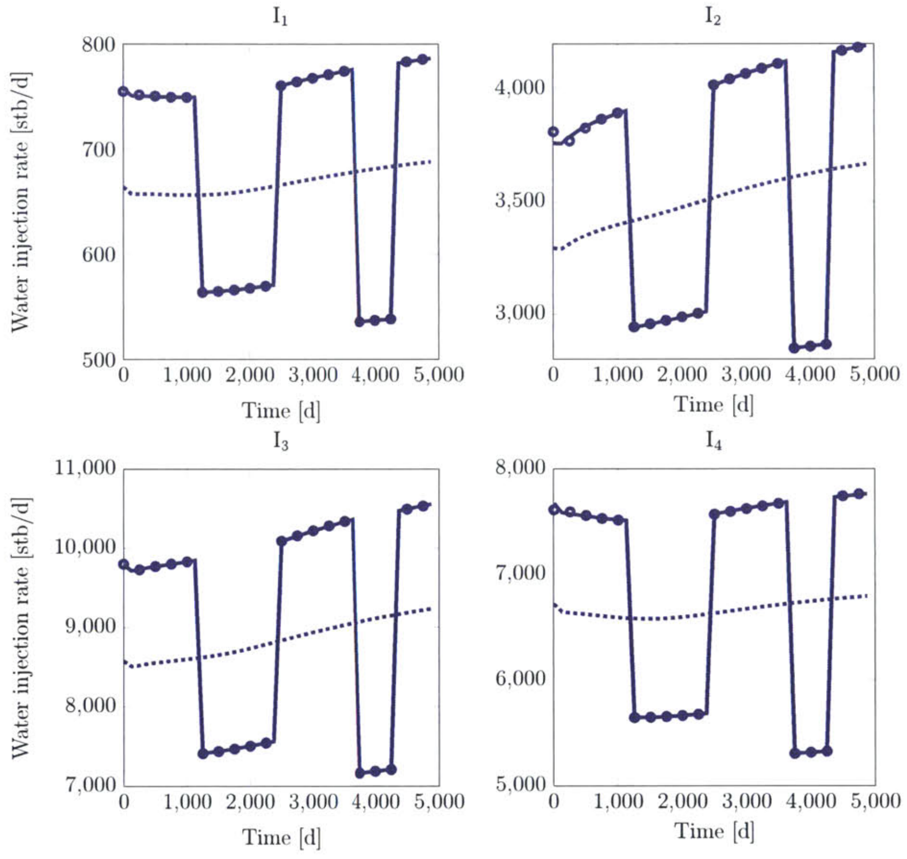


Figure 4.7: Injection flow rates from Pierce model. Dashed line represents training run trajectories, circles indicate ROM results using testing schedule, and solid line represents the reference testing run using FOM.

4.1.3 Well Placements Optimization

In Chapter 3, we introduced the MCS method, and its hybrid counterpart MCS-MADS to solve the well placement problem. The well placement problem usually imposes a vast change in boundary conditions that necessitates the use of FOMs. However, instead of solely relying on FOMs to achieve optimal well placements, which incurs a relatively high computational cost, our approach utilizes ROMs to locally scrutinize the search space and improve the optimization outcome, while introducing a minor increase in computational cost. In this framework, the FOM is used as the primary engine to globally search the solution space using MCS while the ROM is incorporated in MADS algorithm to locally improve the optimization results. This multi-scale approach entails the use of ROM in areas of solution space where changes in well locations do not constitute a significant change in production response (e.g., areas with low to moderate water production). Our approach continuously assesses the accuracy of ROM using (3.8) to ensure accurate results are always attained. In case of discrepancy between ROM and the reference FOM (due to significant change in fluid dynamics), a new set of orthogonal bases are generated using a new training set (root-point) and hence a new ROM is reconstructed to incorporate the new changes.

We aim in this example to determine the optimal well placements that maximize undiscounted NPV. The optimization problem presented in (3.1) is expressed as:

$$\begin{aligned} \min_{\mathbf{u}\in U} \quad & J(\mathbf{p},\mathbf{u}) = -\mathrm{NPV}(\mathbf{p},\mathbf{u}) = -r_o Q_o(\mathbf{p},\mathbf{u}) + c_{wp}Q_{wp}(\mathbf{pu}) + c_{wi}Q_{wi}(\mathbf{p},\mathbf{u}), \\ \text{subject to} \quad & \mathbf{g}(\mathbf{p},\mathbf{u}) = 0, \quad \mathbf{c}(\mathbf{p},\mathbf{u}) \leq 0. \end{aligned} \tag{4.1}$$

Here r_o is the price of oil (\$/STB), c_{wp} and c_{wi} are the cost of handling produced water and the cost of injected water (\$/STB) respectively. All prices/costs are held constant in this study. Q_o, Q_{wp} and Q_{wi} are the cumulative oil production, water production and water injection (all in STB) respectively.

Two cases are presented. In the first case, MCS and MCS-MADS methods are applied to determine the optimal well placements in the presence of only bound constraints. The second case additionally incorporates nonlinear constraints. These constraints include the minimum well-to-well distance, minimum oil flow rate, and maximum liquid flow rate. Table 4.1 summarizes key optimization parameters and also the nonlinear constraints. In all cases, the production time-frame is 5,000 days and the production rate controls are kept constant throughout as we apply a sequential optimization strategy. The total number of optimization variables for both cases is 18 (nine areal location variables).

Table 4.1: Simulation and optimization parameters for Pierce model run

Injection BHP	8000 psi
Production BHP	1500 psi
r_o	\$100/STB
c_{wp}	\$10/STB
c_{wi}	\$5/STB
Minimum oil production rate	2000 STB/day
Maximum liquid production rate	6000 STB/day
Minimum well-to-well distance	1000 m
Maximum water cut	0.66

In the standalone MCS, a population size of 70 nests is used, implying a maximum of 70 function evaluations per MCS iteration. A $2n$-polling stencil is used for the MADS algorithm. This constitutes a maximum of $2n$ function evaluations per MADS iterate ($2n = 36$ in this case). The MADS algorithm stops if the poll size is below 0.1 and the MCS method stops after ten generations. We note here that all iterates and their corresponding function evaluations are saved in cache memory, and hence simulation runs to evaluate some iterates are avoided if these iterates have been visited previously. This is in general a good practice as metaheuristic algorithms such as MCS tend to revisit solutions as the solution converges.

Case 1: Bound Constraints Only

For this case, the nonlinear field rates and well-to-well minimum distance constraints are neglected. The well locations are bounded within the field boundaries. Considering the stochastic nature of the applied algorithms, and the fact that the optimization solution surface may contain multiple optima, each of these methods is repeated five times starting from five different initial guesses. The NPVs for the five initial guesses, together with their mean $E(NPV)$ and standard deviation (σ), are shown in Table 4.2. The first guess contains well locations depicted in Figure 4.1. The remaining guesses are randomly generated from a uniform distribution within the bounds of the problem.

Table 4.2: NPVs corresponding to five initial guesses used in Pierce model optimization runs (best is bold)

Run #	Initial guess NPV [$ MM]
1	**418**
2	176
3	352
4	236
5	297
$E(NPV)$	296
σ	85

The optimization results for both MCS and the hybrid MCS-MADS are displayed in Figure 4.8 and summarized in Table 4.3. Figure 4.8 shows the evolution of NPV versus the number of simulations (averaged over five runs) for each method. Both methods achieve significant improvement over the mean initial guess NPV. We clearly see that the hybrid MCS-MADS outperforms the standalone MCS in terms of both higher E(NPV) and lower σ. Table 4.3 presents the final optimal NPVs for the five runs and the best NPV for each method is in bold.

Figure 4.8 accentuates the superiority of the hybrid MCS-MADS over the standalone MCS. Both methods are designed to be able to avoid poor local optima—through Lèvy flights. However, MCS-MADS also includes local search ability that improves the optimization results, evident by the higher mean objective function in Figure 4.8 and the relatively smaller σ in Table 4.3.

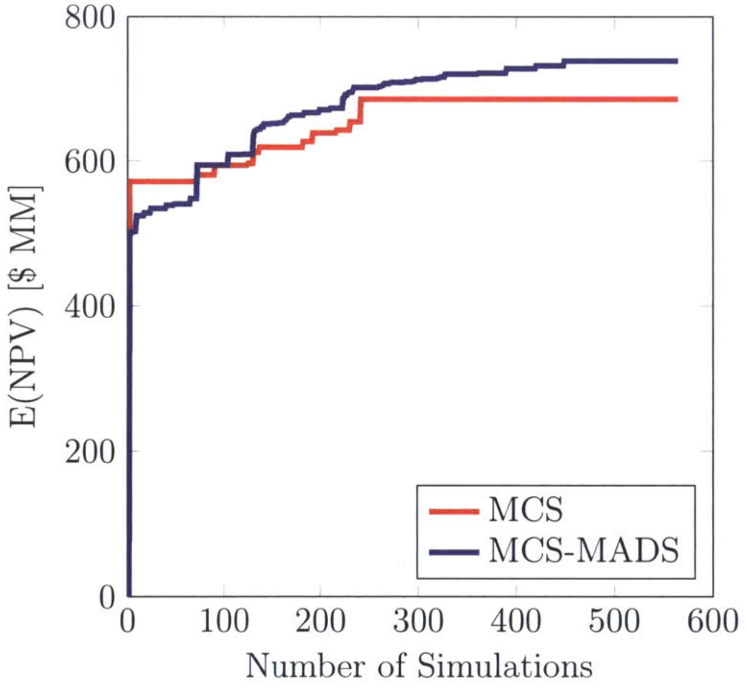


Figure 4.8: Evolution of mean NPV for bound constraints in Pierce model (Case 1). The MCS-MADS algorithms improves the average NPV by 8% (from $685 MM to $738 MM).

Table 4.3: Final NPVs corresponding to five guesses used in Pierce model optimization runs for Case 1 (best is bold)

Run #	MCS NPV [$ MM]	MCS-MADS NPV [$ MM]
1	**718**	**745**
2	654	726
3	694	745
4	673	742
5	687	734
$E(NPV)$	685	738
σ	21	7

Figure 4.9 depicts the evolution of NPV using ROM at the end of each MADS iteration (within the MCS-MADS algorithm) against that of FOM. Initially, we see less agreement between NPV obtained using ROM (blue curve) and FOM (red curve) as the used initial poll size Δ_0^p is relatively large. Poll size is accordingly adjusted for subsequent iterations to achieve a better overall agreement and hence remain close to the ROM's root-point. Figure 4.10 displays the progress of the calibration parameter κ during MCS-MADS optimization. Here ($\bigcirc$) indicates accepted MADS iterations where $\epsilon \leq \kappa$, while ($\times$) represents rejected iterations. MADS iterations are mainly rejected when the suggested optimization solution is far from the ROM's root-point. We observe that most of the rejected iterations have negative calibration parameter ($\kappa < 0$). This occurs when the ROM achieves an opposite progress to that of the FOM (i.e., the objective function value obtained using ROM decreases from iteration k to $k+1$ while the objective function value obtained using FOM increases from iteration k to $k+1$). The main advantage of using ROM in this work is to be able to efficiently guide the optimization algorithm toward an optimal solution. Therefore, we can tolerate low ROM accuracy as long as we follow the FOM (true) ascent direction. This is illustrated by the low ϵ cut-off (dashed line) in Figure 4.10.

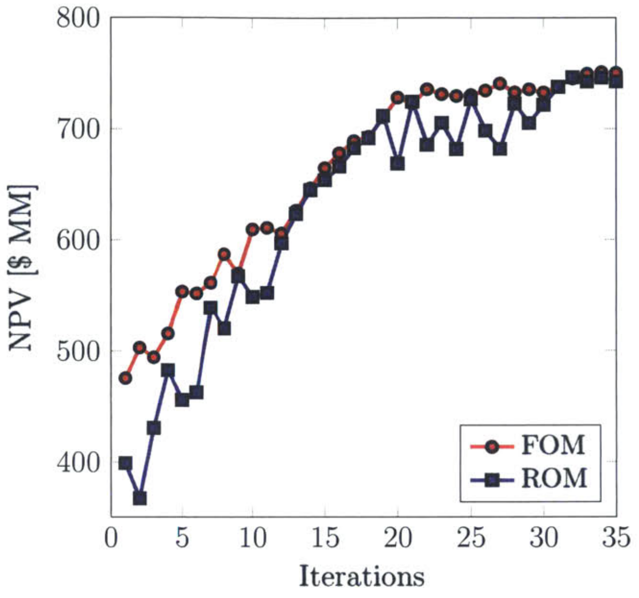


Figure 4.9: Evolution of NPV for bound constraints (run # 1) at the end of each MADS iteration showing the effect of ROM update during optimization to keep the solution from ROM close to that of FOM (Pierce model).

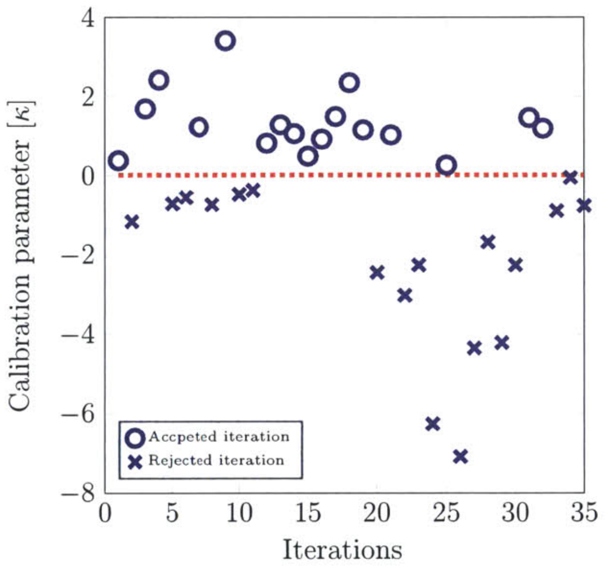


Figure 4.10: Calibration parameter at the end of each MADS iteration (Case 1) for bound constraints from Pierce model run. MADS iterations are rejected when the suggested optimization solution is far from the ROM's root-point

The MCS-MADS algorithm in this case requires an additional computational cost that is equivalent to 40 FOM simulation runs over the standalone MCS, which requires a total of 700 FOM runs. This added cost is associated with approximately 650 ROM runs during polling (equivalent to 5 FOM runs) and 35 FOM runs required for accuracy assessment. Thus, the MCS-MADS algorithm adds only 6% computational cost while increasing the average NPV by 8% (from \$685 MM to \$738 MM). It is evident that the incorporation of ROM in the MCS-MADS algorithm for local search is capable of finding better solutions at a very low additional computational cost.

Case 2: Nonlinear Constraints

We now present a case that incorporates the nonlinear constraints listed in Table 4.1. These constraints include the minimum well-to-well distance, minimum field oil production rate, and maximum field liquid production rate. The total number of nonlinear constraints is thus 38 constraints (36 of which are well-to-well distance constraints). While the distance constraints may be computed prior to the simulation run, the rate constraints may not because the relationship between well locations and production rates are nonlinear (requires nonlinear evaluations using the simulator). We implement the filter approach described in Chapter 3 in conjunction with both MCS and MCS-MADS to handle these constraints. We again run each algorithm five times using the same initial guesses specified in Case 1.

Table 4.4 summarizes the optimization results for each method and Figure 4.11 depicts the progress of mean NPV for the feasible solutions. The curves for the two methods appear after approximately 130 function evaluations (simulation runs) as all initial guesses we start with lead to infeasible solutions—solutions that violates the aggregated nonlinear constraint. MCS-MADS approach is observed to provide superior results relative to its standalone counterpart method. Note here that the best achieved optimal NPV in this case is lower than that in Case 1. This is reasonable as constraints may prohibit solutions that lead to higher overall NPV but with excessive liquid production.

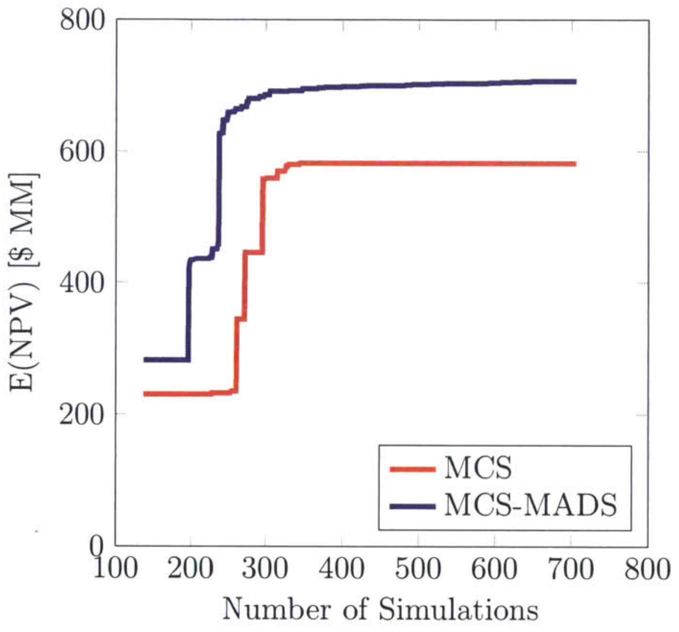
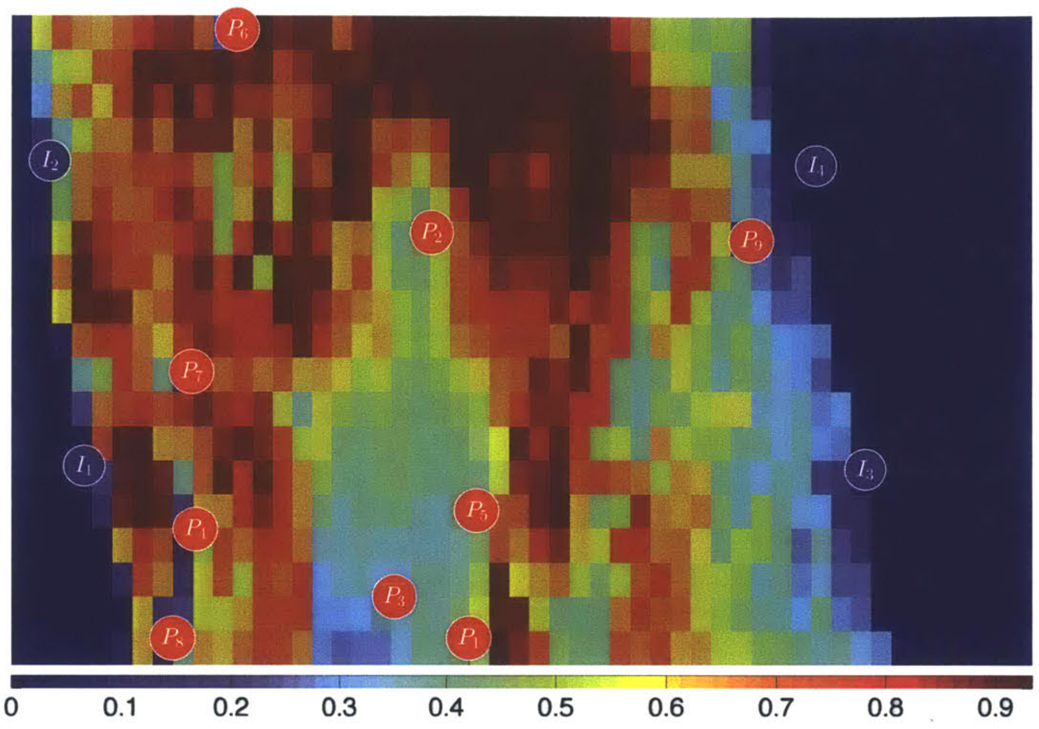


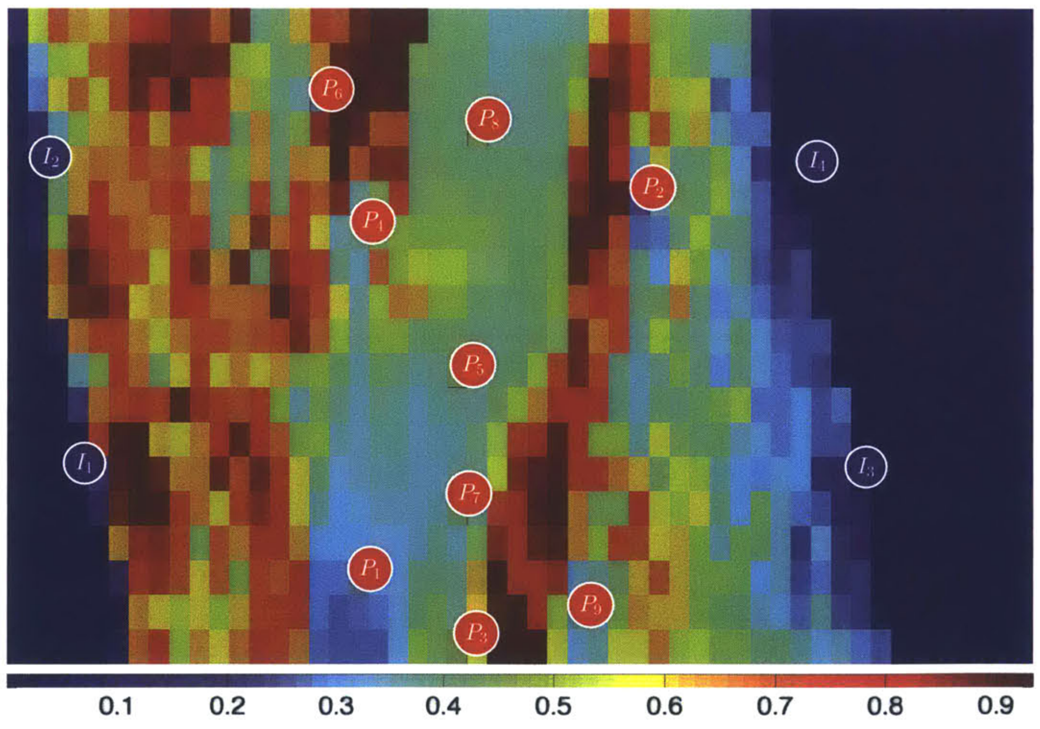
Figure 4.11: Evolution of mean NPV for nonlinear constraints (Case 2). The MCS-MADS algorithms improves the average NPV by 21%.

Table 4.4: Final NPVs corresponding to five initial guesses used in Pierce model optimization runs for Case 2 (best is bold)

Run #	MCS NPV [$ MM]	MCS-MADS NPV [$ MM]
1	564	715
2	548	675
3	560	**732**
4	**672**	695
5	564	711
$E(NPV)$	582	706
σ	46	22



(a) Optimal well locations from MCS

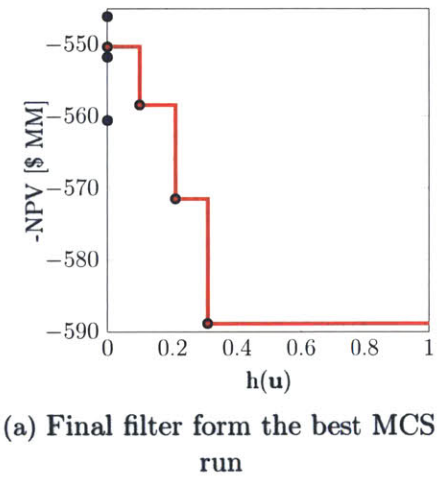


(b) Optimal well locations from MCS-MADS

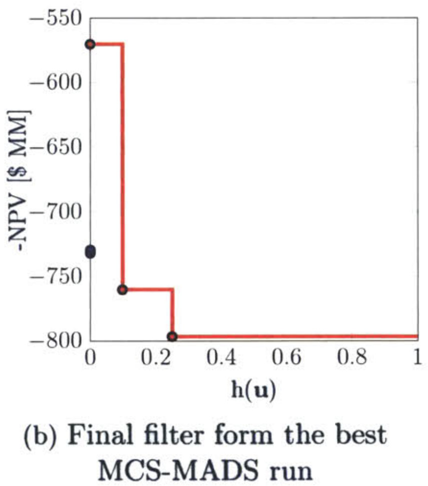
Figure 4.12: Optimal well locations displayed on the final oil saturation of the top layer of Pierce model (red indicates oil). This shows higher volumes of oil produced using well placements suggested by the MCS-MADS algorithm.

Figure 4.12 displays the final oil saturation map of the top layer in addition to the optimal well locations of the best solutions for both methods. It is clear that the optimal well locations from MCS-MADS contribute to higher volumes of displaced oil as opposed to MCS. This is partly because the majority of the wells are located in areas of thicker oil column (middle of the field). In addition, the rest of the wells are placed in the east flank where the injection rate is higher due to lower initial pressure.

The final filters for both MCS and MCS-MADS (run # 3) are depicted in Figure 4.13. The red points represent the infeasible non-dominated solutions that define the filter while the blue points represent the feasible solutions. The filter measures the sensitivity of the objective function $J(\mathbf{p}, \mathbf{u})$ to the constraint violation $h(\mathbf{u})$—similar analogy to the pareto frontier in multi-objective optimization. It quantifies how much gain can be realized in the objective function as a result of relaxing the corresponding constraints. If we relax the aggregate constraint violation by about 25% in Figure 4.13(b), we would be able to achieve a NPV of $796 MM (an additional 9%).



(a) Final filter form the best MCS run



(b) Final filter form the best MCS-MADS run

Figure 4.13: Final filter for optimal well placement in Pierce model (run # 3). Blue circles represent feasible solutions while red circles represent the non-dominated infeasible solutions. The filter envelope from MCS-MADS is closer to the feasibility axis.

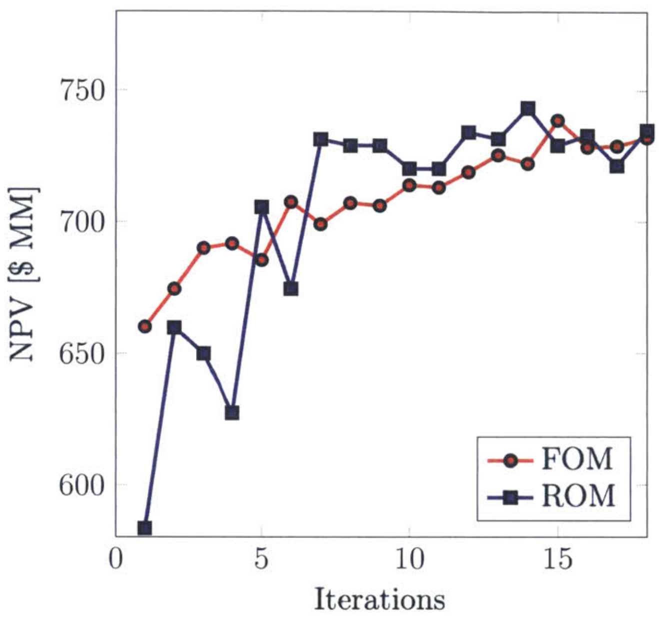


Figure 4.14: Evolution of NPV for nonlinear constraints (run # 3) at the end of each MADS iteration showing the effect of ROM update during optimization to keep the solution from ROM close to that of FOM (Pierce model).

Figure 4.14 depicts the evolution of NPV using ROM at the end of each MADS iteration (within the MCS-MADS algorithm) against that of FOM in the presence of constraints for the best run (run # 3). A total of 18 MADS iterations have been completed in this run. Consistent with observations from Case 1, we see continuous improvement in the quality of the results achieved using the ROM. This is due to the adjustment in the poll size to remain close to the root-point in addition to improving the quality of the ROM by accumulating more snapshots and hence constructing richer orthogonal basis. Figure 4.15 displays the progress of the calibration parameter κ at the end of each MADS iteration. A total of three iterations have been accepted in this case as opposed to 17 in Case 1 which indicates the increase level of difficulty due to the presence of constraints. Even though the percentage of accepted iterations is relatively low (17% compared to 49% in Case 1), the effect of local search attained by MADS is very significant as it improves the solution quality produced by the MCS algorithm.

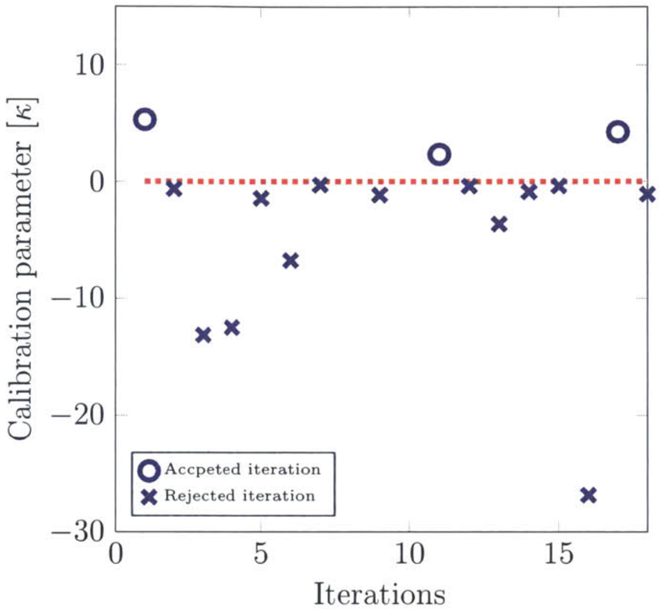


Figure 4.15: Calibration parameter for nonlinear constraints (Case 2) at the end of each MADS iteration (Pierce model). Here (◯) indicates accepted MADS iterations where $\epsilon \leq \kappa$, while ($\times$) represents rejected iterations. MADS iterations are mainly rejected when the suggested optimization solution is far from the ROM's root-point

In this example, the MCS-MADS algorithm improves the mean NPV by approximately 21%. This significant increase in NPV incurs an additional computational cost of only 3% (equivalent to a total of 21 FOM runs). This exemplifies the benefit of using the hybrid MCS-MADS approach over the standalone MCS especially when improvement in NPV is very significant compared to the associated increase in computational cost.

4.1.4 Well Controls Optimization

We next search for the optimal well controls using optimal well locations attained in previous section using MCS-MADS. Recall that well placements were achieved using a constant (time-invariant) BHP. Although this approach—referred to as sequential approach—is suboptimal as shown by [61], we expect nonetheless an increase in the overall NPV after finding optimal well controls.

We perform the optimization run using an initial guess in which all injection well BHPs are set to 8000 psi while all production well BHPs are set to 1500 psi (similar controls to those used during the well placement optimization case). We aim to maximize the NPV using (4.1) under well water cut constraint. In this case, producers are not allowed to produce water cut higher than 0.66. The constraint is aggregated using (3.5) and we implement the optimization run using both ROM (guided and updated using FOM) and standalone FOM.

Table 4.5 details the progress of the optimization run. The main model used during this optimization run is the ROM and therefore we expect a gradual increase in the NPV obtained from ROM ($J_k^{\mathcal{R}}$) at the end of every trust-region step. FOM is used at the end of each trust-region step in order to assess the quality of the ROM results and update the orthogonal basis to construct a new ROM (using a new root-point). A total of twelve trust-region steps are performed, out of which seven steps are accepted. The calibration parameter ρ indicates how well the ROM results follow that of the FOM. We are interested in the accuracy of the ROM as an absolute objective function value (NPV) at the end of each trust-region run as well as the direction of the objective function provided by the ROM—with higher emphasis on the latter. A negative calibration parameter, (i.e., steps 4,6,8, 10, and 12) designates an opposite progress in terms of NPV between the ROM and the FOM. Taking the fourth step as an example, the ROM indicates an increase in NPV from \$834 MM to \$853 MM, while the FOM shows a decrease from \$860 MM to \$806 MM. Clearly, the ROM promotes a set of well controls that result in lower NPV. In this case, the trust-region step is rejected and repeated with a smaller trust-region radius as a higher degree of deviation from the root-point may have occurred. A positive calibration parameter indicates that both the ROM and the FOM yield an increase in NPV (similar ascent direction). The magnitude of the calibration parameter determines the accuracy of the ROM. In this example, we set the lower limit for the calibration parameter to 0.1. This means that we can tolerate a less accurate ROM result as long as it follows the same NPV direction provided by the FOM.

Table 4.5 also lists the aggregate constraint violation at the end of each trust-region step. Overall, we see that the ROM yields feasible solutions (examined by the FOM) excepts for the first step. This demonstrates the ROM ability to solve nonlinear constraint problems, in addition to the robustness of the filter approach to achieve feasible solutions at the end of each step.

Table 4.5: Progress of NPV and nonlinear constraint at the end of each trust-region step for ROM and FOM (Pierce model)

k	ρ	$J_k^{\mathcal{R}}$	$h_k^{\mathcal{R}}$	J_k	h_k
#		[$ MM]		[$ MM]	
1	0.66	738	0	695	1.45
2	1.32	837	0	826	0
3	5.67	834	0	860	0
4	-5.40	853	0	806	0
5	0.14	886	0	866	0
6	-1.56	895	0	852	0.0031
7	51.00	903	0	867	0
8	-0.22	912	0	865	0
9	2.67	918	0	907	0
10	-2.00	936	0	871	0
11	3.62	939	0	983	0
12	-3.92	977	0	834	0

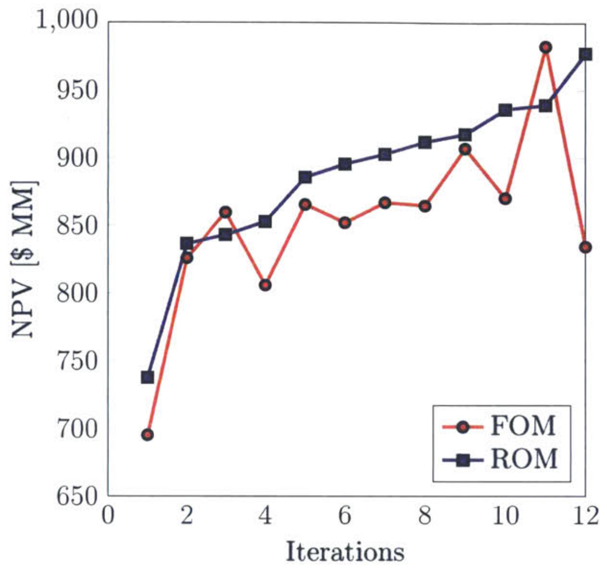


Figure 4.16: Evolution of NPV at the end of each trust-region step for optimal well controls showing the effect of trust-region algorithm to keep ROM solutions close to those obtained using FOM (Pierce model).

Figure 4.16 depicts the evolution of NPV using ROM at the end of each trust-region step against that of FOM in the presence of constraints. We observe here the effect of updating the trust-region radius on the ROM results. In multiple instances, the trust-region framework is able to successfully change the direction of the objective function and recover the accuracy of the ROM through adjusting the trust-region radius and remaining within the vicinity of the root-point. Figure 4.17 visually depicts changes in trust-region radius during the optimization. We generally observe a declining trend in the trust-region radius especially toward the end of the optimization run as we approach the optimal well controls.

Figure 4.18 shows the water cut trajectories corresponding to the optimal solution, together with the constraint limit listed in Table 4.1. We observe that all wells honor the imposed maximum water cut limit. The constraint is active for well P_6 and the water cut trajectory is bound to violate the constraint except that higher BHP control value is imposed by the algorithm for subsequent time steps.

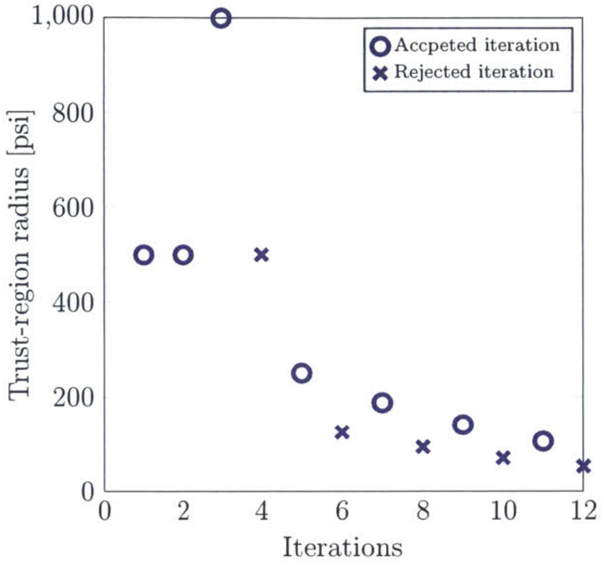


Figure 4.17: Evolution of NPV for optimal well controls at the end of each trust-region step (Pierce model).

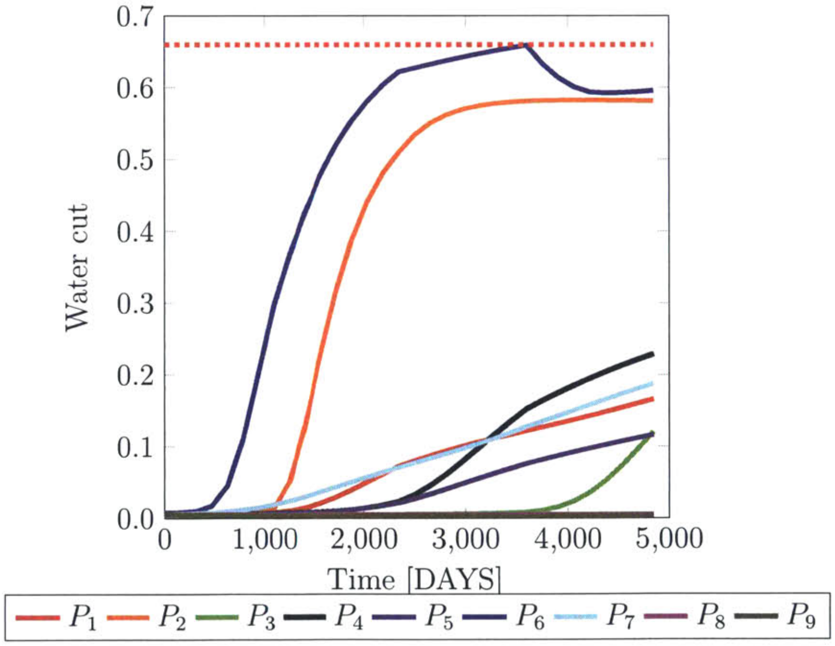


Figure 4.18: Well water cut for optimal well controls in Pierce Model showing that all wells satisfy the imposed constraint. Red dashed line indicates constraint limit.

Figure 4.19 shows the progress of the NPV when the FOM is solely used as the main objective function evaluation engine and compares it to the case when the FOM is guided by the ROM (guided FOM). The guided FOM achieves a slightly higher NPV of $983 MM while the standalone FOM achieves $979 MM. Figures 4.20 displays the optimal set of controls for the injection wells whereas Figure 4.21 displays the optimal controls for the production wells for both cases. The optimal solutions display different characteristics which is expected as the use of ROM alters the actual objective function surface and attracts different optimal solution.

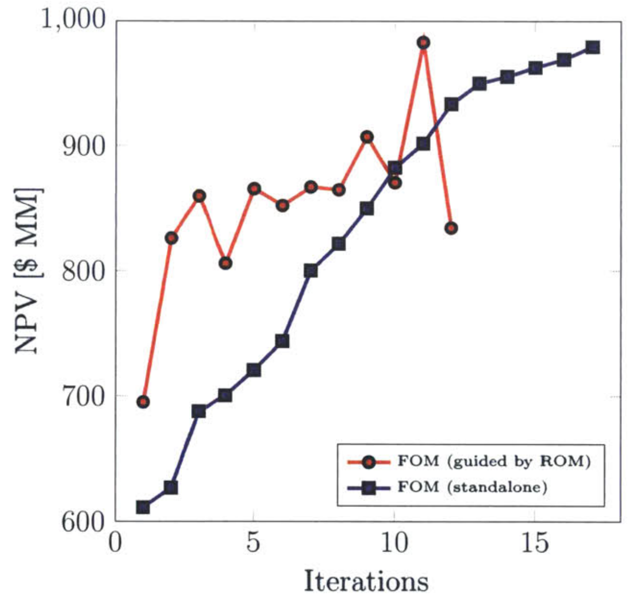


Figure 4.19: Comparison of the evolution of NPV for optimal well controls at the end of each trust-region step using standalone FOM and FOM guided by ROM (Pierce model).

The FOM entails a total of 1,250 simulation runs, while the runtime for the ROM is equivalent to only 47 FOM simulation runs (including FOM assessment runs). This exemplifies the ROM's capability to provide feasible solution with substantial computational reduction. Thus, computational resources can be better utilized to explore more of the solution space.

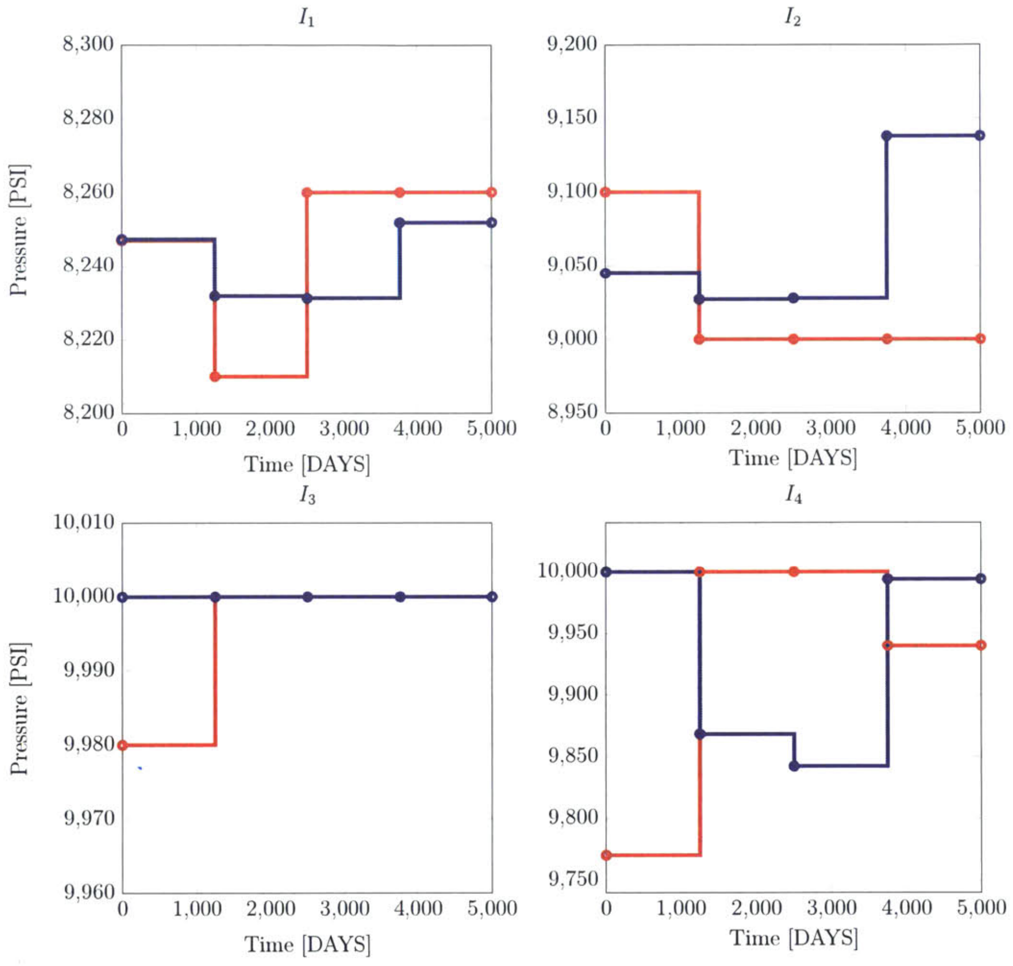


Figure 4.20: Optimized BHP profiles for injection wells (Pierce model). Differences in optimal control schedules exist as ROM provides an approximate solution.

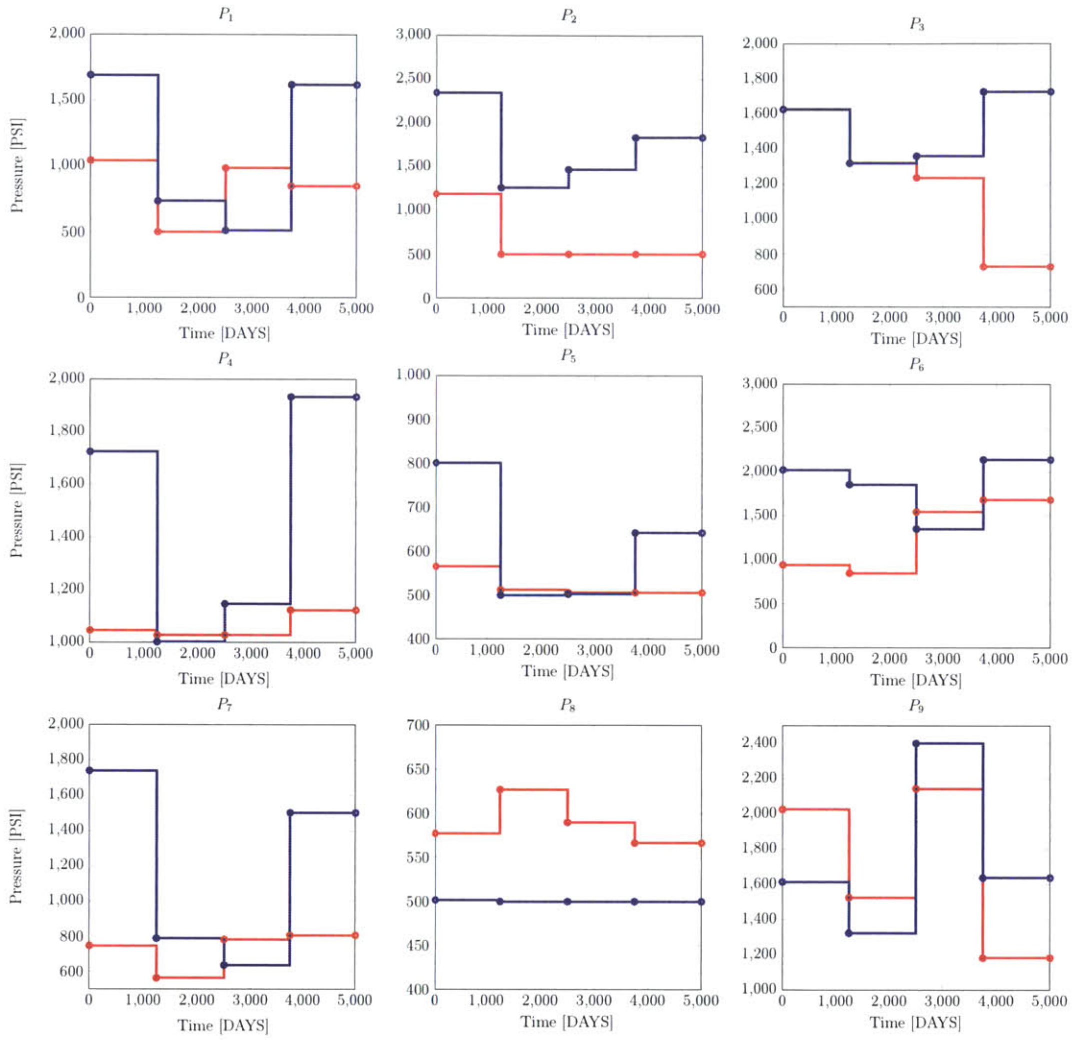


Figure 4.21: Optimized BHP schedule for production wells (Pierce model). Differences in optimal control schedules exist as ROM provides an approximate solution.

4.2 Stata Model: SPE-10 Model (top 5 layers)

4.2.1 Description of the Simulation Model

The reservoir model employed in this example, depicted in Figure 4.22, represents the top five layers of the SPE 10th comparative study [34]. The model contains a total of 66,000 grid blocks ($N_x = 60, N_y = 220, N_z = 5$). The physical dimension of each grid block is 20 ft $\times$ 10 ft $\times$ 20 ft. There are four horizontal production wells (designated P_1–P_4) initially located at the corners of the model and two injection wells (designated I_1–I_2) drilled in the middle. Production wells are completed in the top layer while injection wells are completed at the bottom most two layers. Production wells are under BHP control while injection wells are injecting water at a constant rate equivalent to 0.7 PVI.

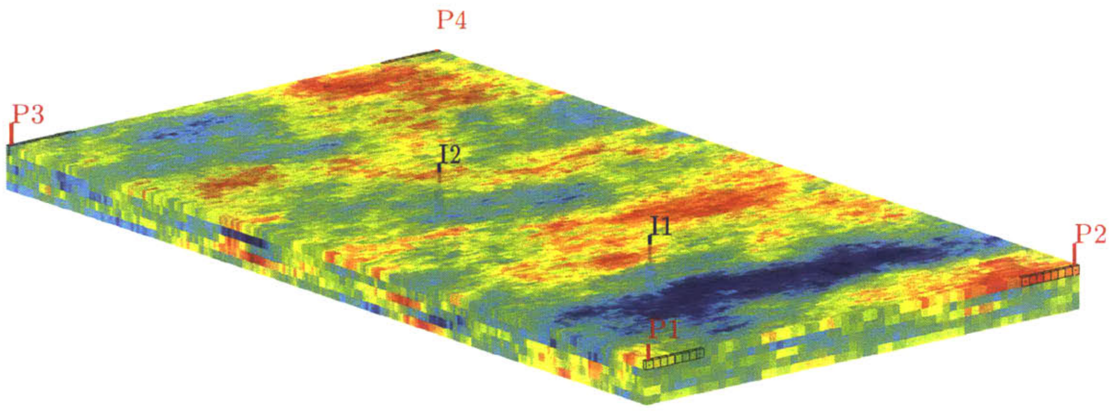


Figure 4.22: Three-dimensional grid structure of Stata model with two vertical injectors and four horizontal producers. ($\log_{10} k_x$ is shown in the background).

Figure 4.23 displays the permeability in the x-direction for each layer. The permeability is taken to be a diagonal tensor, with $k_x = k_y$. The porosity is constant in this model and set equal to 0.25. The initial water and residual oil saturations are zero. For oil, we set $\rho_o = 45$ lb/ft^3, $\mu_o = 3$ cp; for water, we set $\rho_w = 65$ lb/ft^3, $\mu_w = 1$

cp. The system is incompressible and capillary pressure is neglected in this example. The relative permeabilities for the oil and water phases are specified using (2.47) and (2.48) with $k_{ro}^0 = k_{rw}^0 = 1$ and $a = b = 2$.

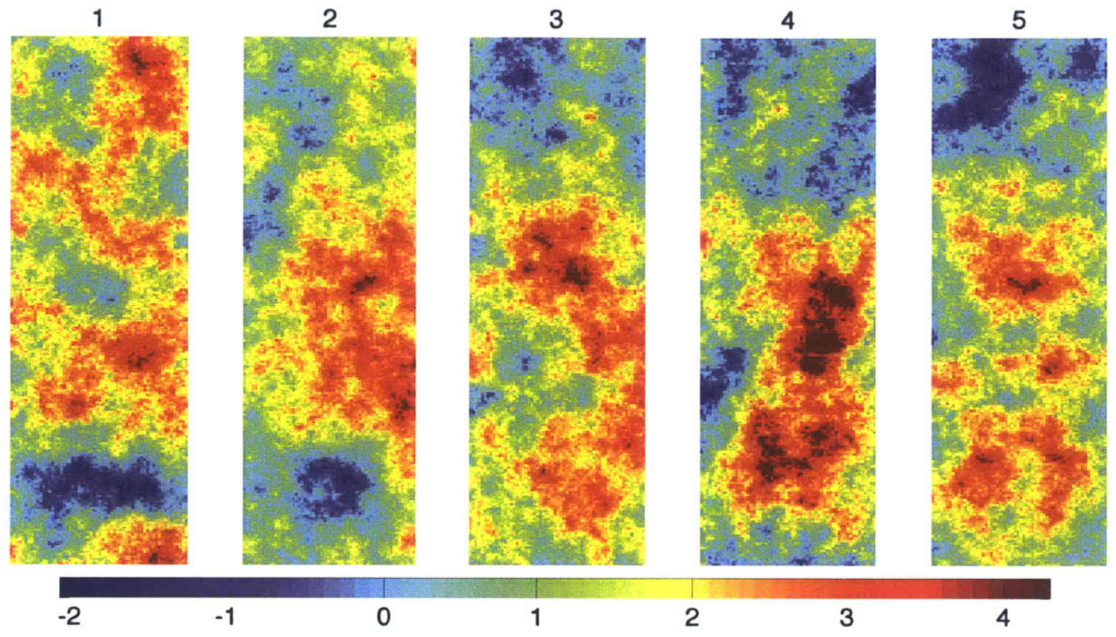


Figure 4.23: Permeability in the x-direction for Stata model ($\log_{10} k$ with k expressed in mD).

4.2.2 Training and Testing Results Using ROM

In order to construct the ROM for this example, a training simulation run is performed using FOM in order to collect solution state (saturation) and nonlinear term (flux) snapshots. During the training simulation, all production wells operate at a constant BHP of 1,000 psi (injection rates are kept constant throughout this example). The training case is simulated for 5,000 days and a total of 320 snapshots are collected and used to build the orthogonal bases. The reduced saturation and flux bases contain 55 and 161 columns respectively. A total of 966 interpolation points are selected using Algorithm 3 (including relational links for upwinding). Figure 4.24 displays the spatial distribution of the flux term interpolation points. We observe in this example that the interpolation points span the whole field, Unlike the Pierce model (Figure 4.4) where they are concentrated in regions of high permeability.

Figure 4.25 displays the test schedule used to assess the accuracy of the ROM. Each BHP control varies between 500 psi and 2,500 psi. It is perturbed every 625 days comprising a total of eight perturbations. Figure 4.26 shows the oil and water production rates from the four production wells. The ROM generally demonstrates close agreement with the reference FOM with substantial reduction in computational costs. In this example, one FOM run is approximately equivalent to 127 ROM runs.

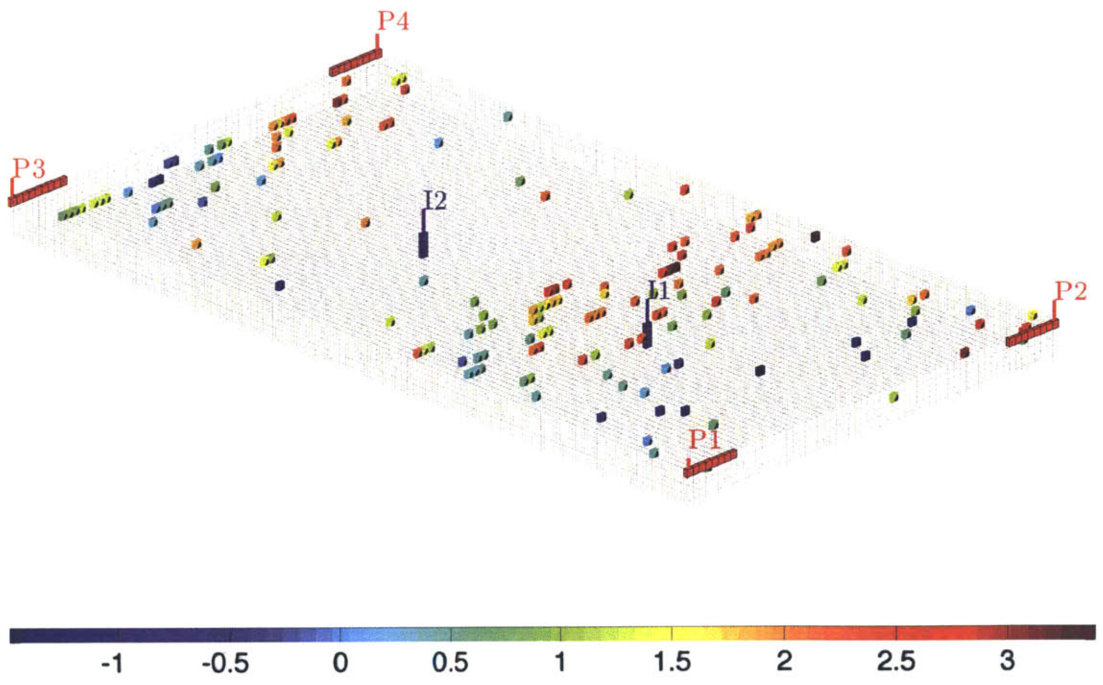


Figure 4.24: Spatial distribution of interpolation points selected to reconstruct flux field for Stata model (background displays $\log_{10} k_x$ for grid blocks determined by the DEIM Algorithm).

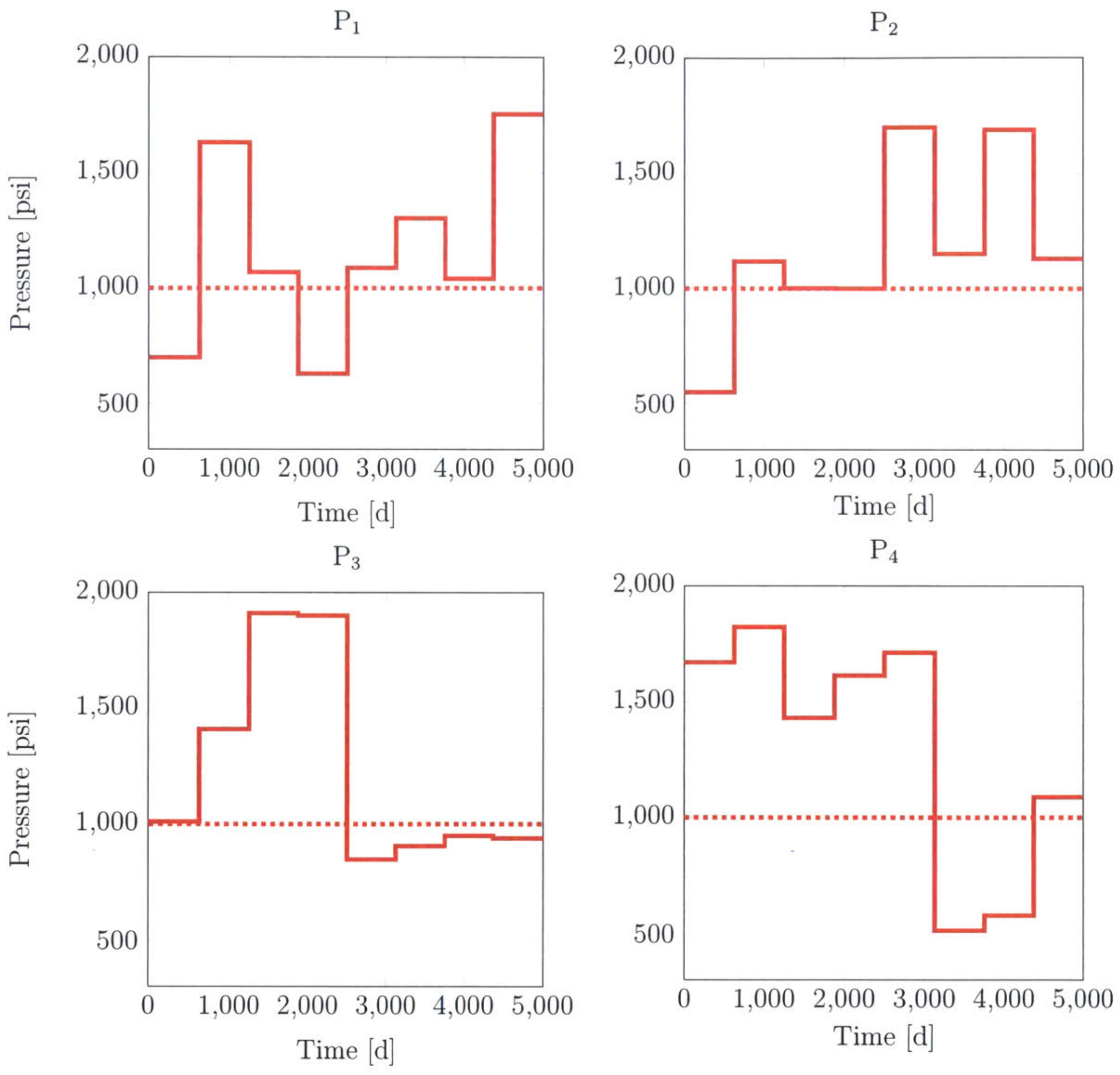


Figure 4.25: Schedules for training and testing cases for the Stata model. Dashed lines represent training BHP controls while solid lines depict testing BHP controls. The training schedule is chosen to be constant while the training schedule is perturbed every 625 days comprising a total of eight perturbations.

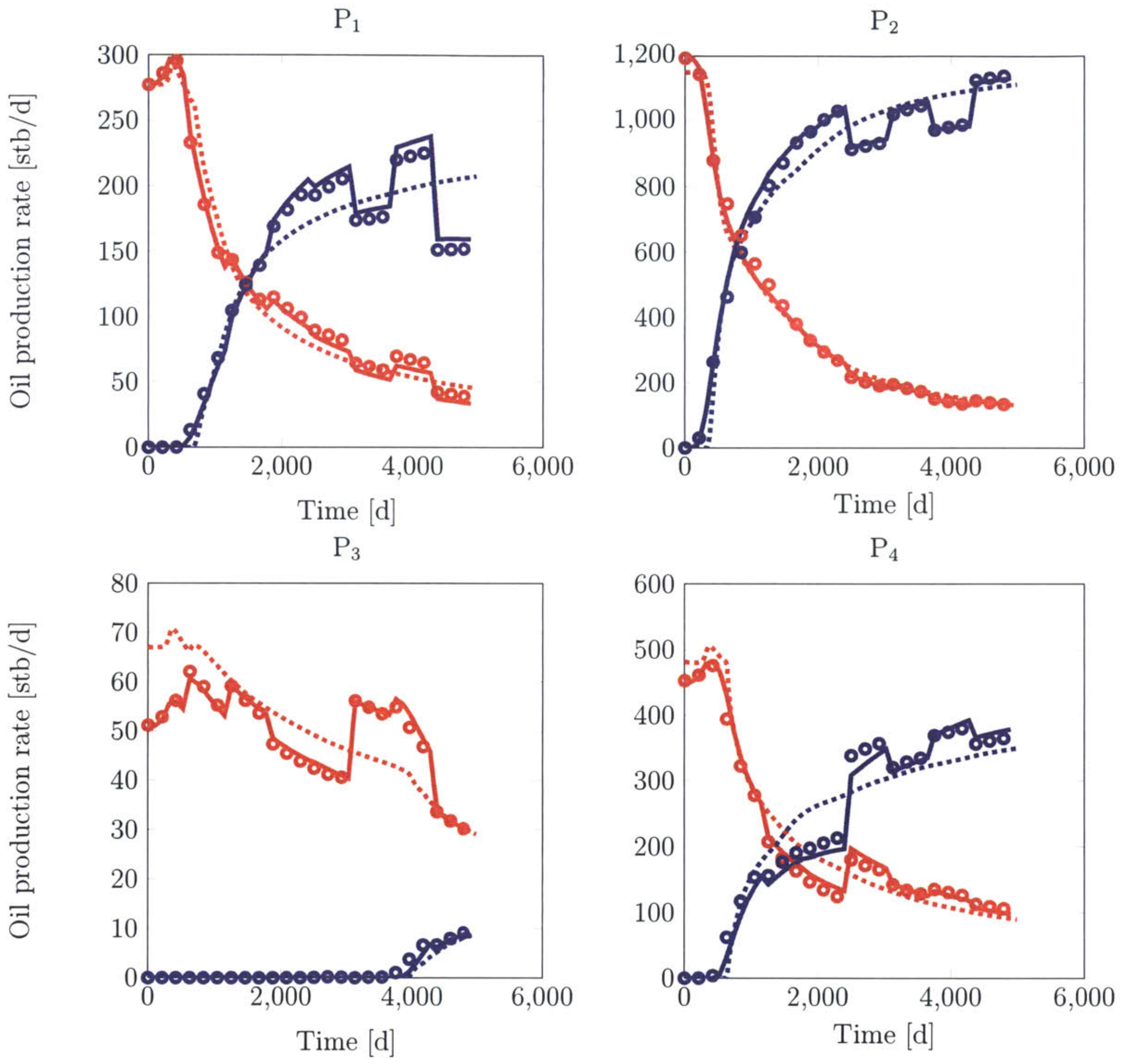


Figure 4.26: Production rates for each well in Stata model. Dashed line represents training run trajectory, circles indicate ROM results using testing schedule, and solid line represents the reference testing run using FOM. Red designates oil while blue designates water.

4.2.3 Well Placements Optimization

In this example, we seek to maximize oil recovery through finding optimal well placements and production controls. This is achieved by minimizing the Lorenz coefficient for well placements (to improve recovery) and maximizing NPV for well controls (to ensure highest economic revenue). The Lorenz coefficient is defined as:

$$L_c = 2\left(\int_0^1 (F(\Phi) - \Phi) d\Phi \right), \tag{4.2}$$

where both the flow capacity F and the storage capacity Φ are calculated using a numerically computed flux field and streamline time-of-flight (TOF). Lorenz coefficient measures the deviation of the $F - \Phi$ diagram from an idealized piston-like displacement. It varies between 0 (homogeneous displacement) to 1 (infinitely heterogeneous displacement) and therefore serves as a unique measure of the flow— or dynamic— heterogeneity. It has been shown by [63] that minimizing Lorenz coefficient results in an optimal volumetric sweep. However, minimizing Lorenz coefficient only does not necessarily result in maximizing field's economic revenue especially at later times when water production is excessive. Thus, we combine Lorenz function with NPV to maximize both recovery and revenue.

Assuming horizontal trajectories remain within a single layer, a horizontal well can be defined using two spatial points, the heel (i_1, j_1) and the toe (i_2, j_2). However, such a definition is not practical as it lacks the ability to predetermine the length of the horizontal section (the well length is specified in practice before commencing drilling process). Therefore, we follow the approach described in [107] where a total of four parameters are used to sufficiently define the location of the well-reservoir contacts with a predetermined horizontal trajectory length. These parameters include the heel coordinates (i_1, j_1), the azimuth angle θ, and the constant horizontal section length l.

Using simple trigonometry, we can then calculate the toe point (i_2, j_2) which is used in addition to the heel point as inputs to the simulator.

The reservoir simulator employed in this work follows a cell-centered approach where reservoir properties and well coordinates are specified at the center of the grid blocks. Therefore, we provide a mapping tool—represented by line-grid intersection algorithm—to transform the well trajectory from the real space (normally a straightline trajectory) to the grid space (staircase trajectory as shown in Figure 4.27).

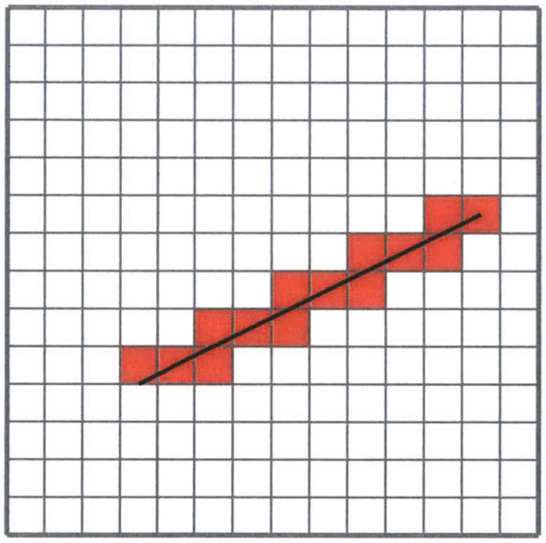


Figure 4.27: Representation of a horizontal well on cell-centered grids showing staircase trajectory.

We aim in this example to find the optimal locations of two vertical injection wells and four horizontal production wells. This implies optimizing a total of twelve discrete areal (i, j) parameters and four continuous angular θ parameters.

Similar to the previous case example, we apply MCS and MCS-MADS methods to determine the optimal well placements. We consider two cases: the first case assumes the presence of only bound constraints while the second case incorporates nonlinear constraints. These constraints, in addition to other key optimization parameters, are summarized in Table 4.6. In all cases, the production time-frame is 5,000 days and the number of optimization variables is 16.

Table 4.6: Simulation and optimization parameters (Stata model)

Injection rate	1000 STB/day
Production BHP	2000 psi
Minimum oil production rate	300 STB/day
Minimum well-to-well distance	10 grid blocks
Maximum well water cut	0.7

The number of nests for the MCS algorithm is 70, and a maximum of $2n$ function evaluations per MADS iteration is used ($2n = 32$ in this case). Both methods are repeated five times starting from different initial guesses. Lorenz coefficients for the five initial guesses, together with their mean $E(L_c)$ and standard deviation (σ), are shown in Table 4.7.

Table 4.7: Lorenz coefficients corresponding to five initial guesses used in optimization runs for Stata model (best is bold)

Run #	Initial guess L_c
1	0.732
2	0.717
3	0.734
4	**0.655**
5	0.824
$E(L_c)$	0.732
σ	0.071

Case 1: Bound Constraints Only

The optimization results for both MCS and the hybrid MCS-MADS are displayed in Figure 4.28 and summarized in Table 4.8. Figure 4.28 shows the evolution of

Lorenz coefficient versus the number of simulations (averaged over five runs) for each method. Both methods achieve significant improvement over the mean initial guess Lorenz coefficient. Again, we clearly see that the hybrid MCS-MADS outperforms the standalone MCS for each single run as indicated in Table 4.8. MCS-MADS improves the mean Lorenz coefficient by 6% over the standalone MCS.

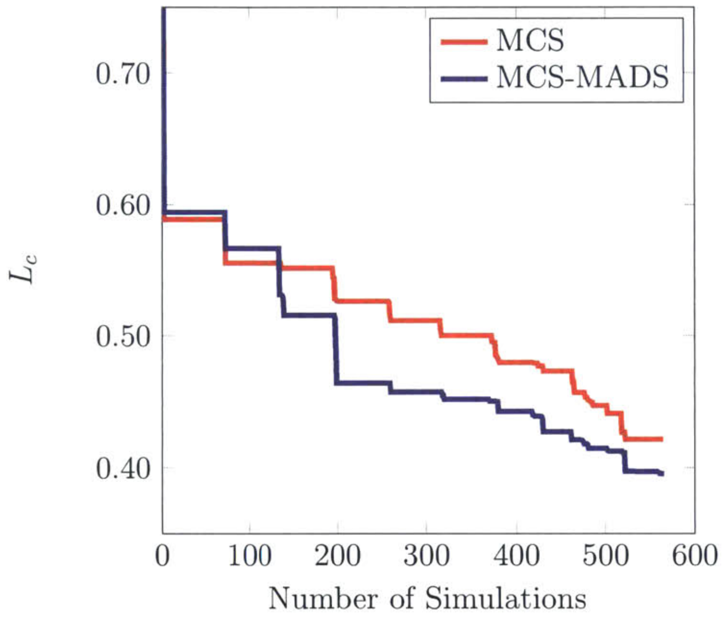


Figure 4.28: Evolution of mean Lorenz coefficient for bound constraints (Stata model). Lorenz coefficient is lowered by 6% using MCS-MADS algorithm.

Table 4.8: Final Lorenz coefficient corresponding to five initial guesses used in optimization run for Case 1 of Stata model (best is bold)

Run	MCS	MCS-MADS
#	L_c	L_c
1	0.411	0.383
2	**0.403**	0.401
3	0.436	**0.381**
4	0.414	0.399
5	0.443	0.411
$E(NPV)$	0.421	0.395
σ	0.017	0.013

Figure 4.29 depicts the evolution of Lorenz coefficient using ROM at the end of each MADS iteration (within the MCS-MADS algorithm) against that of FOM for the best run (run # 3). Close agreement between the FOM and ROM is generally demonstrated which exemplifies the robustness of the ROM in this example. Figure 4.30 displays the progress of the calibration parameter κ during the optimization. Eight MADS iterations are rejected in this case (out of 14 iterations) due to negative calibration parameter.

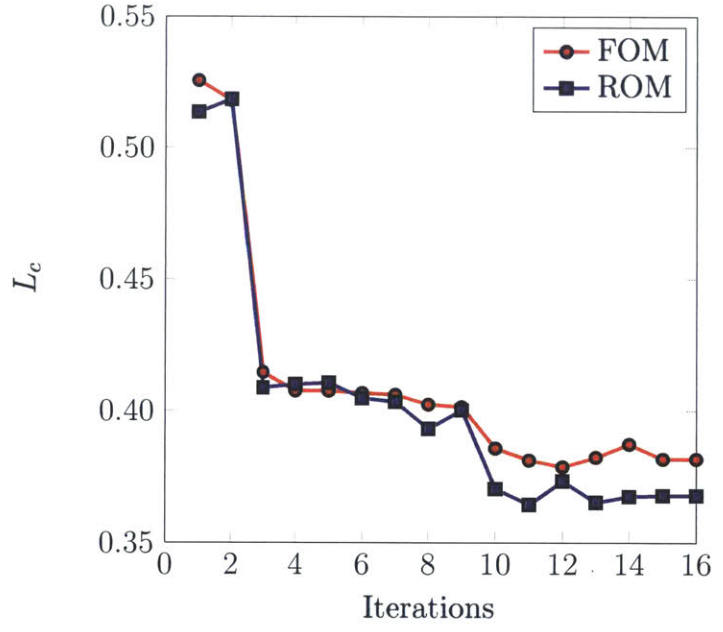


Figure 4.29: Evolution of Lorenz coefficient for bound constraints (Case 1) at the end of each MADS iteration showing the effect of ROM update during optimization to keep the solution from ROM close to that of FOM (Stata model).

Figures 4.31–4.35 display the final oil saturation map for each layer in addition to the optimal well locations. Although both methods share similar global characteristics (injection wells are placed at the northern part of the field while production wells are placed in the south), we see how MCS-MADS better positions producers to target the bypassed oil. We observe in Figure 4.31(a) that P_1 and P_2 are placed close to each other which results in competitive drainage at late simulation times (due to the absence of well-to-well distance constraints). In contrast, MADS-MCS in Figure 4.31(b) better places P_2 in order to target larger volumes of the bypassed oil.

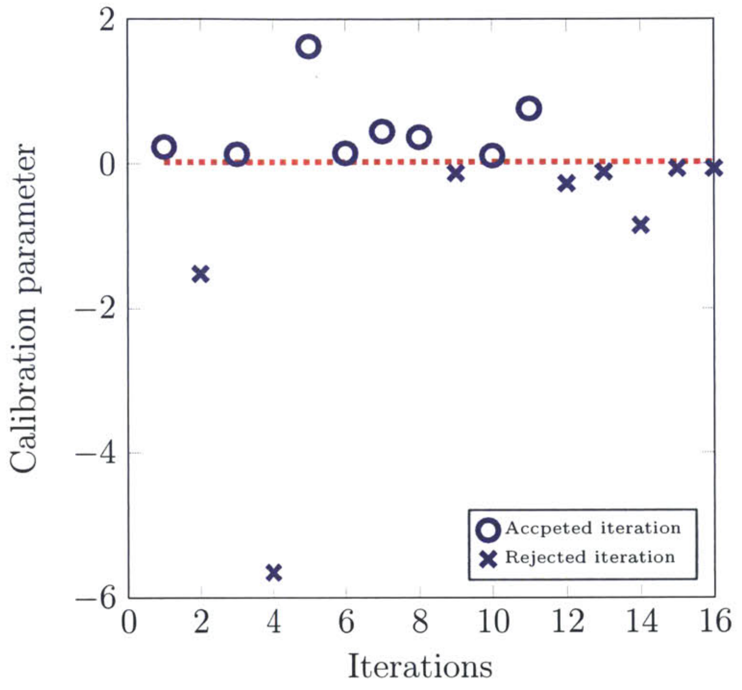
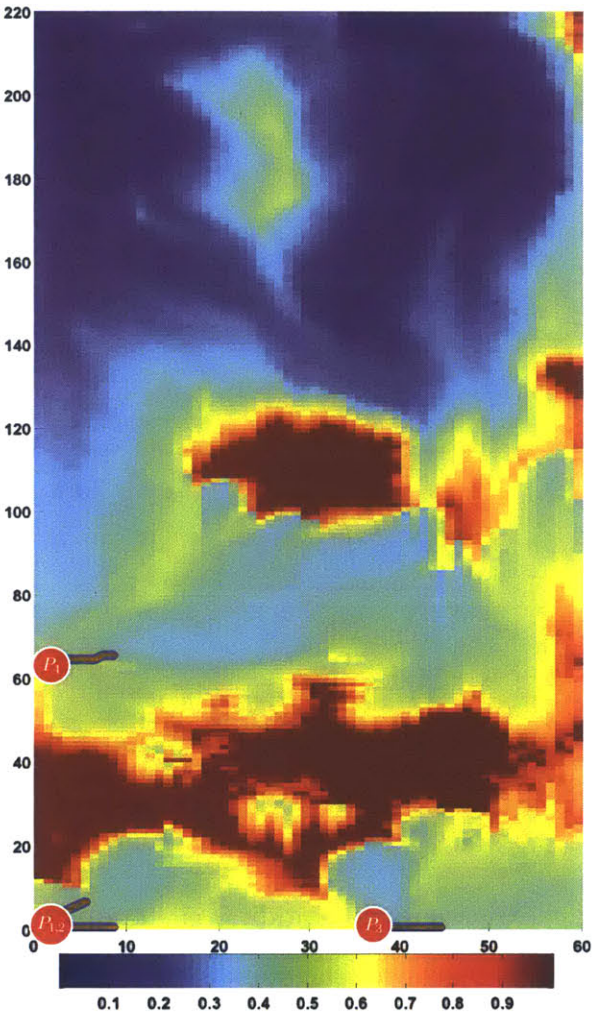
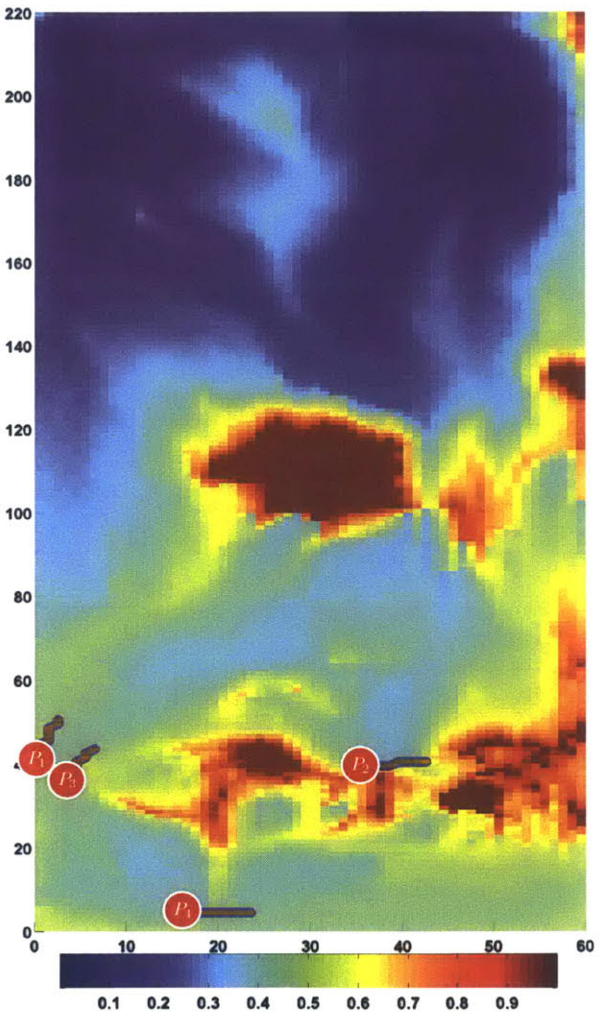


Figure 4.30: Calibration parameter at the end of each MADS iteration for Stata model, bound constraints (Case 1). The dashed line indicates the calibration parameter value cut-off

The hybrid MCS-MADS algorithm in this case required an additional computational cost that is equivalent to 21 FOM simulation runs over the standalone MCS ,which required a total of 700 FOM runs. This added cost is associated with approximately 430 ROM runs during polling (equivalent to 4 FOM runs) and 17 FOM runs required for accuracy assessment and training. Thus, the MCS-MADS algorithm adds only 3% computational cost while lowering the average Lorenz coefficient by 6% (from 0.421 to 0.395).

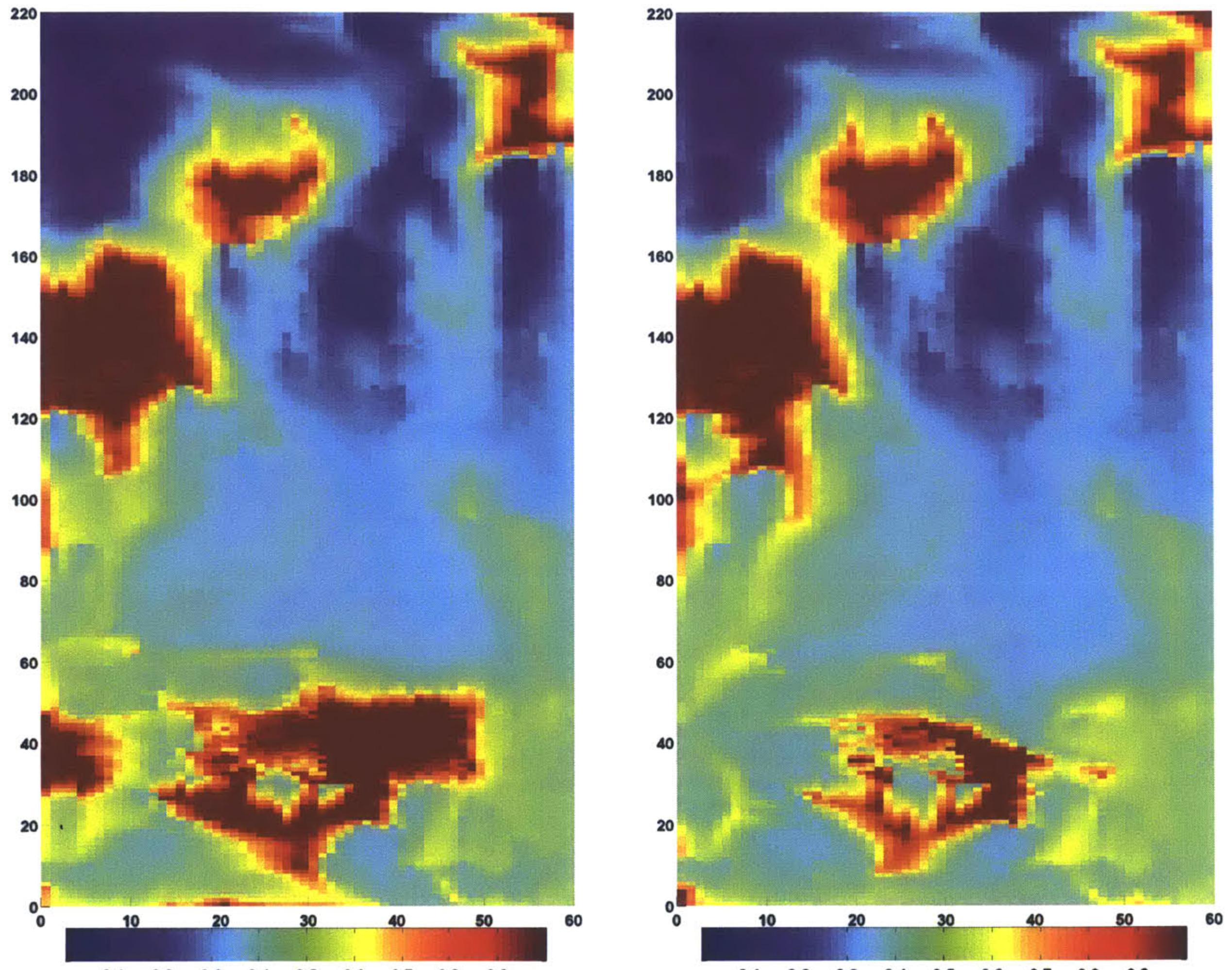


(a) Optimal well locations from MCS



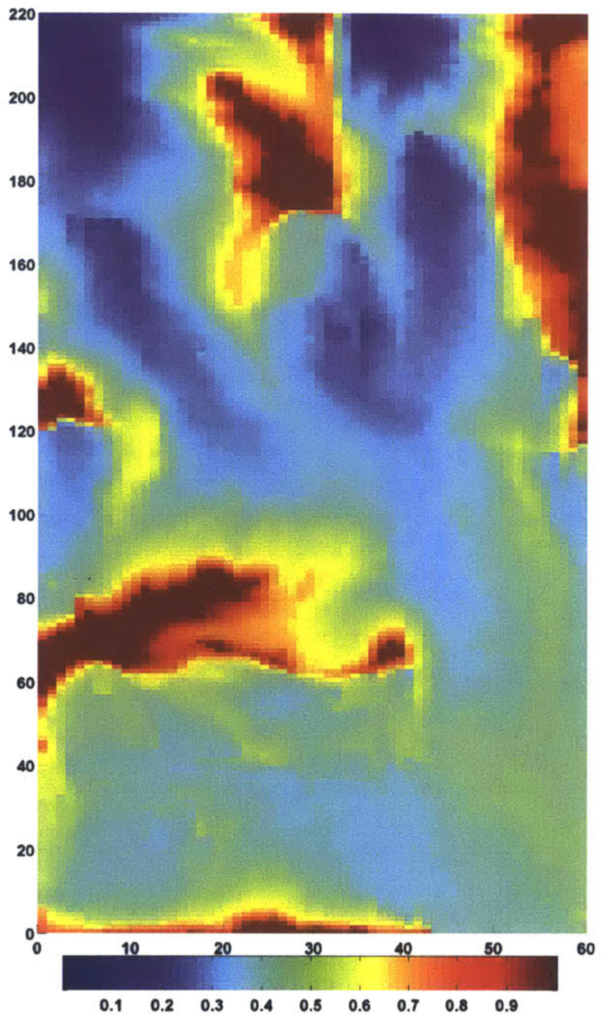
(b) Optimal well locations from MCS-MADS

Figure 4.31: Optimal well locations displayed on final oil saturation map of layer one of Stata model for Case 1. MCS-MADS (right) results in better placement of P_2 which targets larger volumes of bypassed oil (red indicates oil while blue is water).

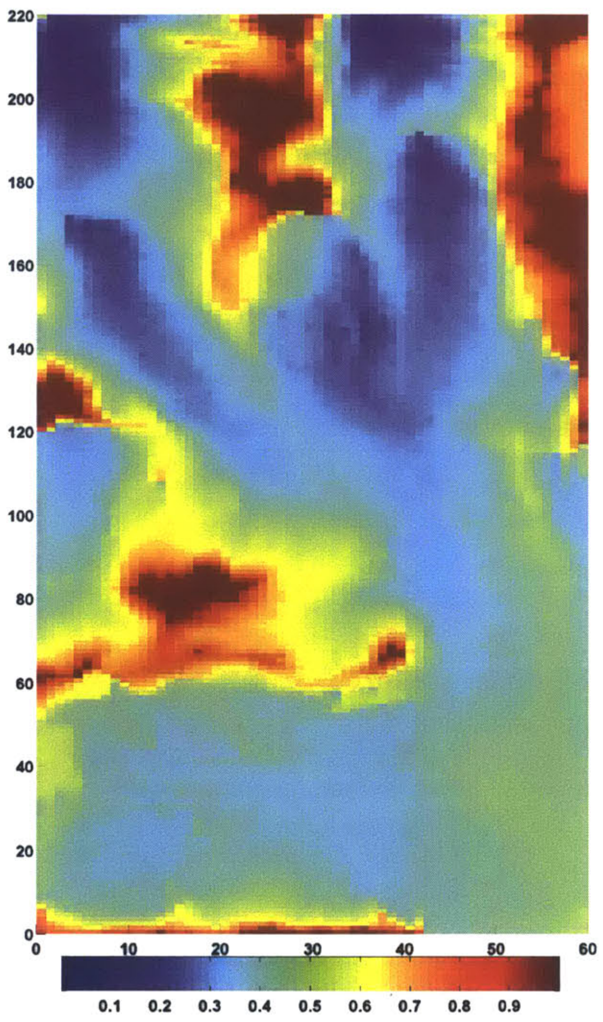


(a) Optimal well locations from MCS (b) Optimal well locations from MCS-MADS

Figure 4.32: Optimal well locations displayed on final oil saturation map of layer two of Stata model for Case 1 (red indicates oil while blue is water). More oil is displaced using well placement from MCS-MADS algorithm.



(a) Optimal well locations from MCS



(b) Optimal well locations from MCS-MADS

Figure 4.33: Optimal well locations displayed on final oil saturation map of layer three of Stata model for Case 1 (red indicates oil while blue is water).

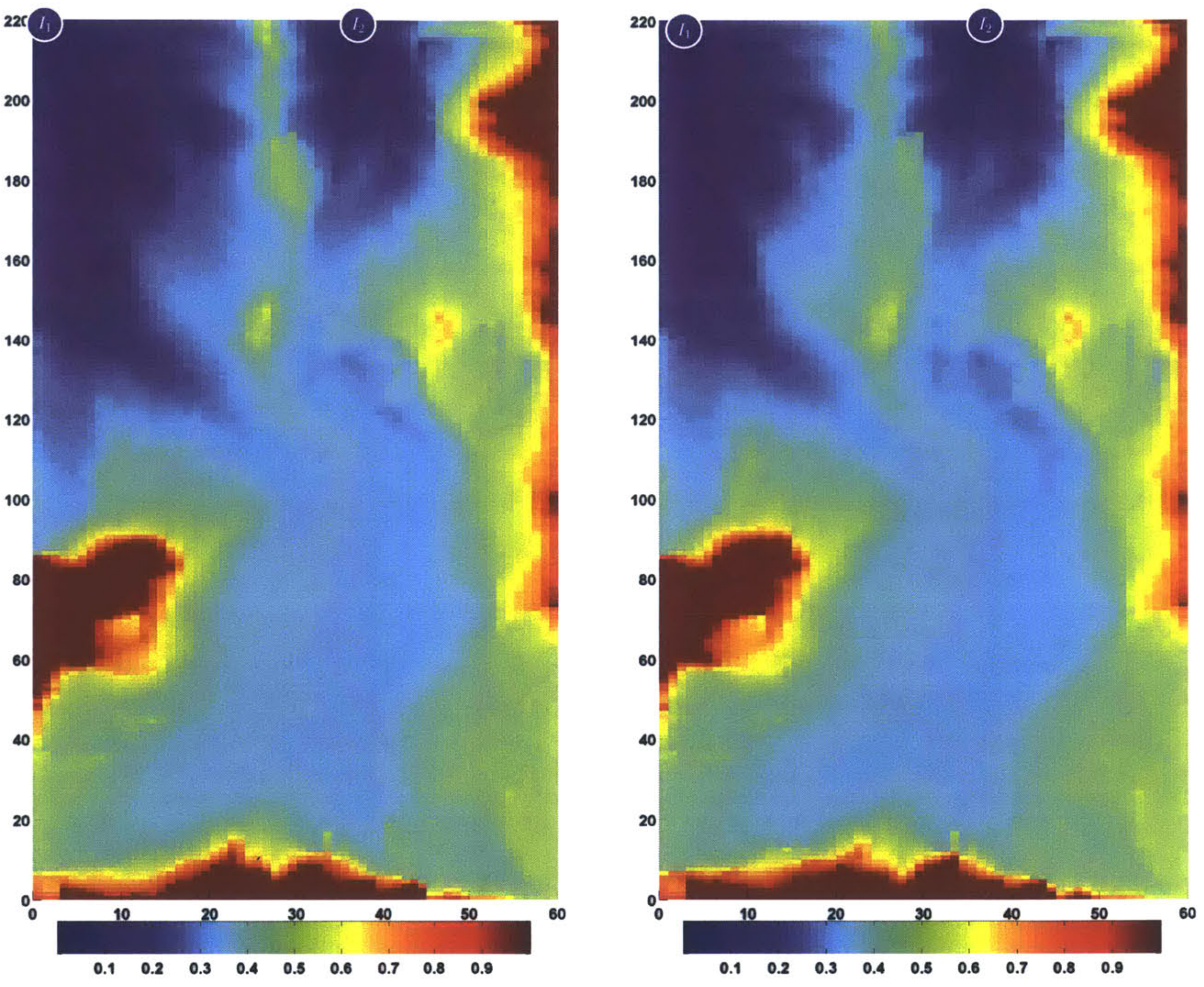
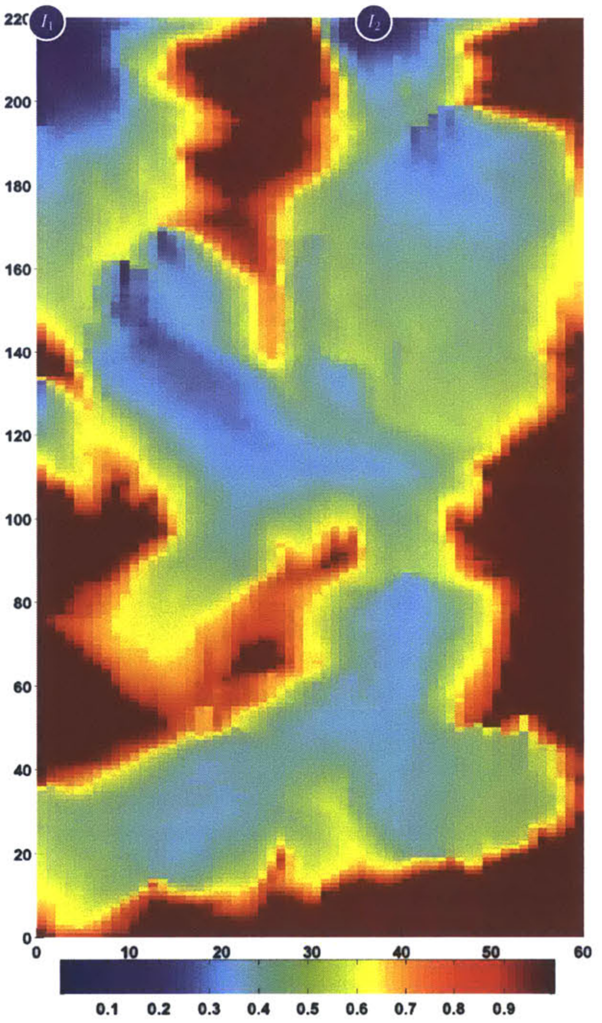
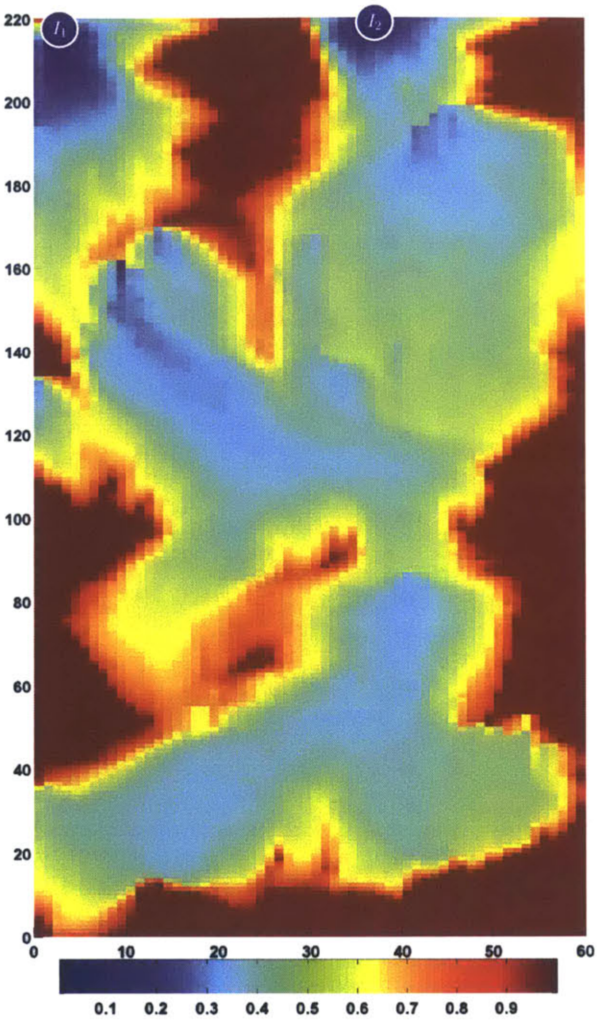

(a) Optimal well locations from MCS
(b) Optimal well locations from MCS-MADS

Figure 4.34: Optimal well locations displayed on final oil saturation map of layer four of Stata model for Case 1 (red indicates oil while blue is water). Both algorithms place injectors at the northern part of the field with small differences due to the local search ability of MCS-MADS algorithm.



(a) Optimal well locations from MCS



(b) Optimal well locations from MCS-MADS

Figure 4.35: Optimal well locations displayed on final oil saturation map of layer five of Stata model for Case 1 (red indicates oil while blue is water). Both algorithms place injectors at the northern part of the field with small differences due to the local search ability of MCS-MADS algorithm.

Case 2: Nonlinear Constraints

We now introduce constraints listed in Table 4.6 to this case. There are a total of 20 constraints (15 of which are well-to-well distance constraints). Table 4.9 summarizes the optimization results for each method. We observe in this case that some initial guesses do not result in feasible solutions. In such a case, the initial guess is most likely far away from any feasible solution (this can be avoided by increasing the number of cuckoo eggs). Although both methods do not yield feasible solutions in Run # 2, we observe that MCS-MADS is able to find a feasible solution in Run # 3. This emphasizes the importance of MADS in our framework as it increases the probability of finding a feasible solution even when starting from a poor initial guess. Figure 4.36 shows the progress of mean Lorenz coefficient for feasible solutions (we include only runs that result in feasible solutions for both methods). MCS-MADS finds its first feasible solution after 340 simulation runs while MCS takes approximately 450 simulation runs. This and the lower mean Lorenz coefficient in Table 4.9 clearly highlights the superiority of MCS-MADS over MCS.

Table 4.9: Final Lorenz coefficient corresponding to five initial guesses used in optimization run for Case 2 in Stata model (best is bold)

Run	MCS	MCS-MADS
#	L_c	L_c
1	**0.413**	**0.390**
2	N/A	N/A
3	N/A	0.433
4	0.429	0.425
5	0.517	0.494
$E(NPV)$	0.453	0.436
σ	0.056	0.043

Figure 4.37 compares the the evolution of Lorenz coefficient computed using ROM at the end of each MADS iteration with that of FOM (Run # 1). Consistent with previous cases, the results obtained via ROM is very close to that of FOM except the last few iterations which are already rejected. A total of 31 MADS iterations have been completed with a success rate of 42%, as shown in Figure 4.38.

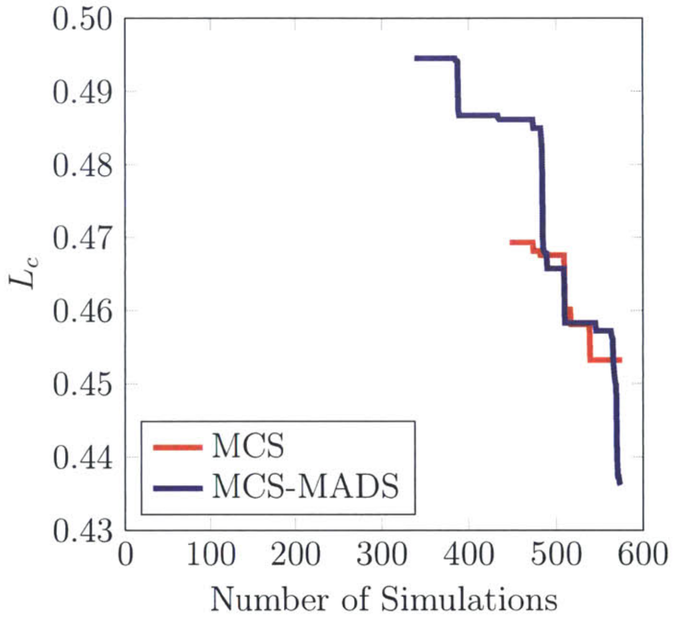


Figure 4.36: Evolution of mean Lorenz coefficient for nonlinear constraints (Stata model). MCS-MADS algorithm finds the first feasible solution after 340 simulations while MCS takes approximately 450 simulations.

Figure 4.39 shows the final filter development for both methods for Run # 1. In this case, we show only the nondominated solutions which define the filter envelope. MCS-MADS provides more rigorous filter as we see more nondominated solutions close to the feasibility threshold. As stated earlier, filters provide detailed analysis of the relationship between objective functions and the corresponding constraints violation. Taking MCS-MADS filter as an example (Figure 4.39(b)), we see that we can further lower the Lorenz coefficient if we relax the aggregated constraint. In particular, we can lower Lorenz coefficient from 0.391 to 0.37 (equivalent to 3.67 \$MM increase in NPV) by relaxing the aggregated constraint to 0.004—For this particular point, the only violated constraint is well P_2 water cut and hence relaxing this constraint from 0.7 to 0.73 would achieve the aforementioned improvement.

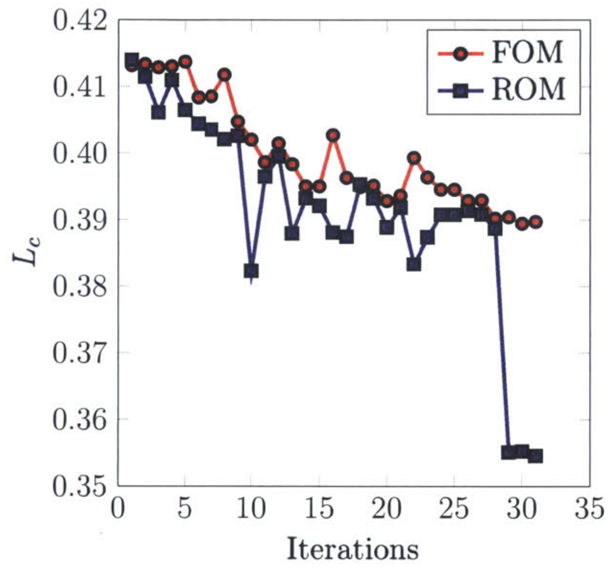


Figure 4.37: Evolution of L_c for nonlinear constraints (run # 1) at the end of each MADS iteration showing the effect of ROM update during optimization to keep the solution from ROM close to that of FOM (Stata model).

Figures 4.40–4.44 display the final oil saturation map in addition to well locations. We again observe that both MCS and MCS-MADS yield similar global solution with MCS-MADS succeeding in targeting bypassed oil with the help of local search.

The hybrid MCS-MADS algorithm in this case required an additional computational cost that is equivalent to 36 FOM simulation runs over the standalone MCS. This added cost is associated with 600 ROM runs during polling (equivalent to 5 FOM runs) and 31 FOM runs required for accuracy assessment. Thus, the MCS-MADS algorithm adds 5% computational cost while lowering the average Lorenz coefficient by 4% (from 0.453 to 0.436).

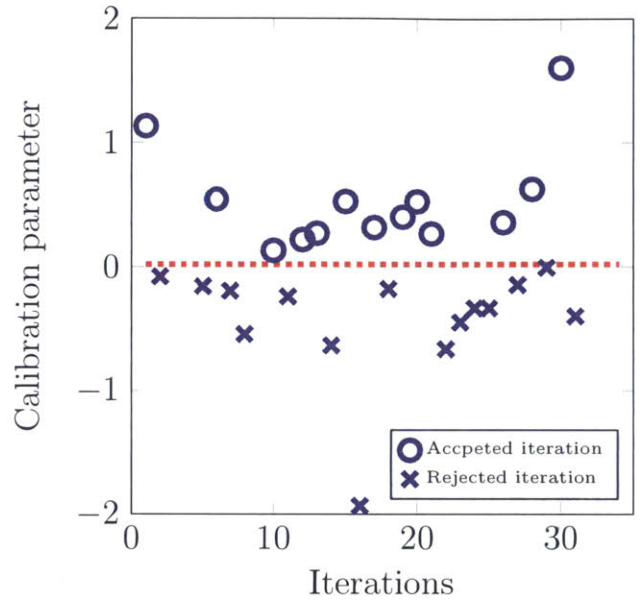
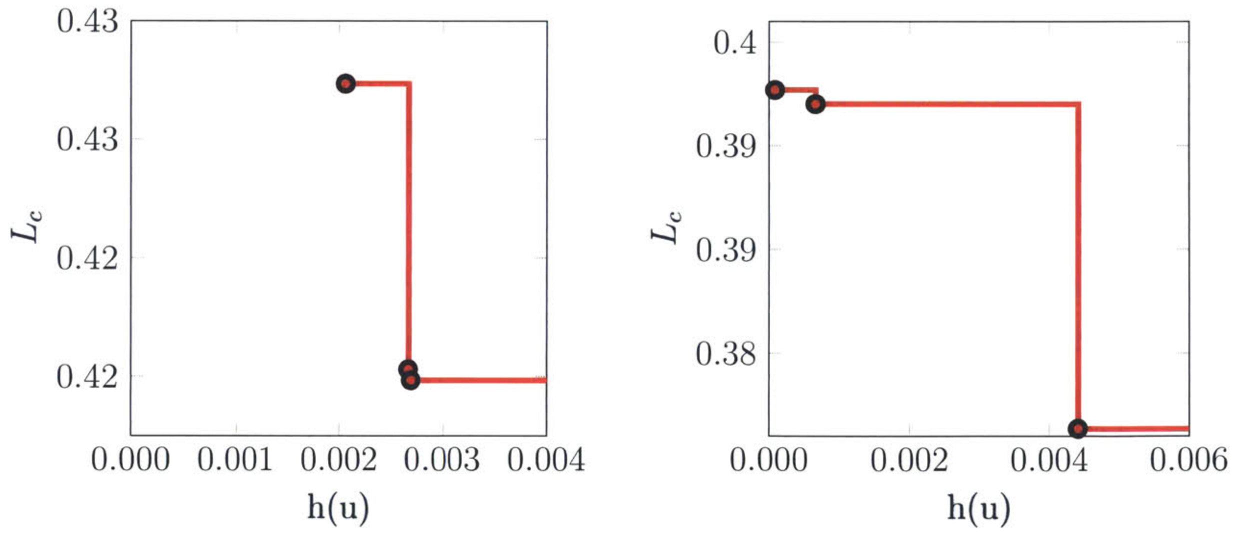
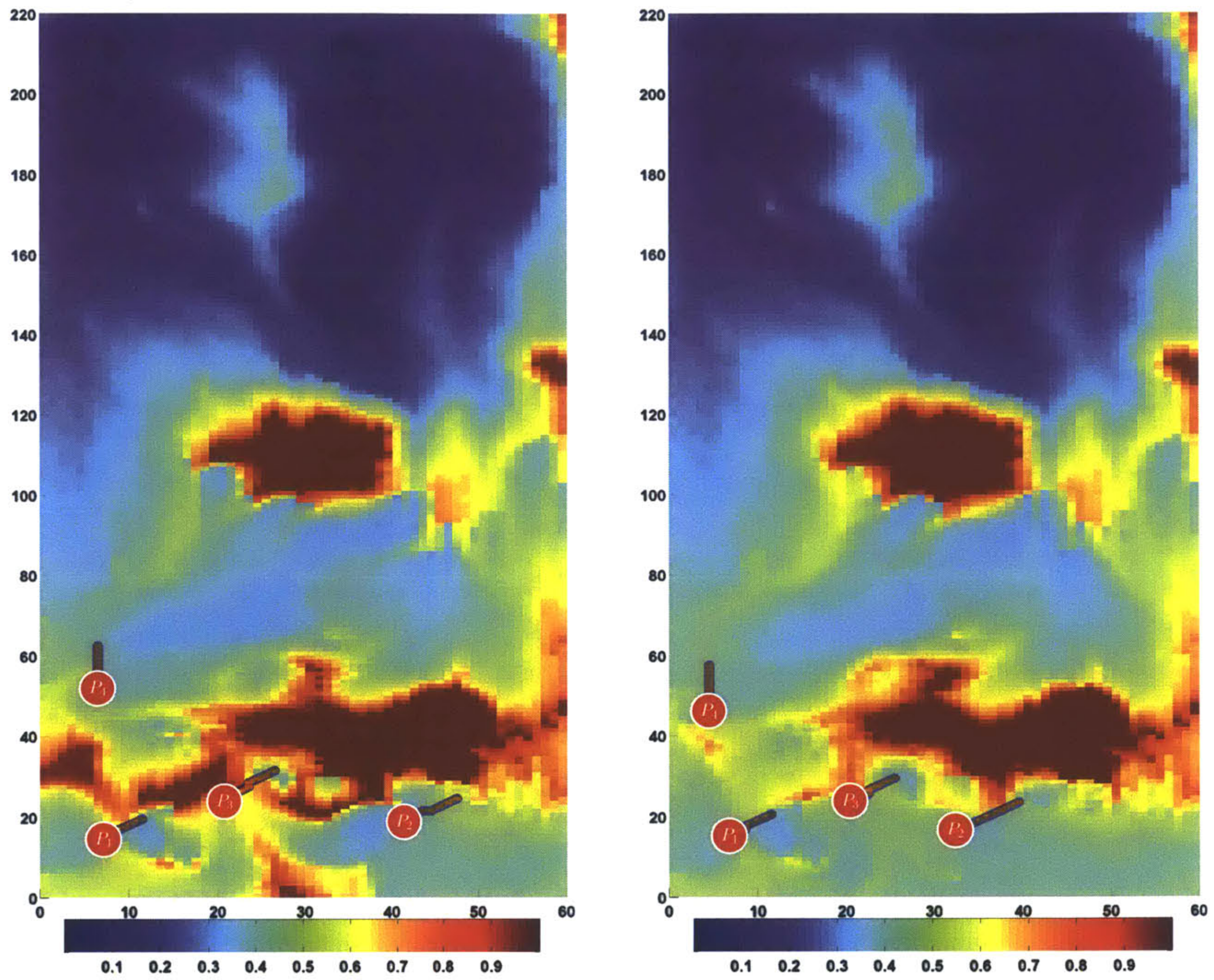


Figure 4.38: Calibration parameter at the end of each MADS iteration for Stata model, nonlinear constraints (Case 2). The dashed line indicates the calibration parameter value cut-off.



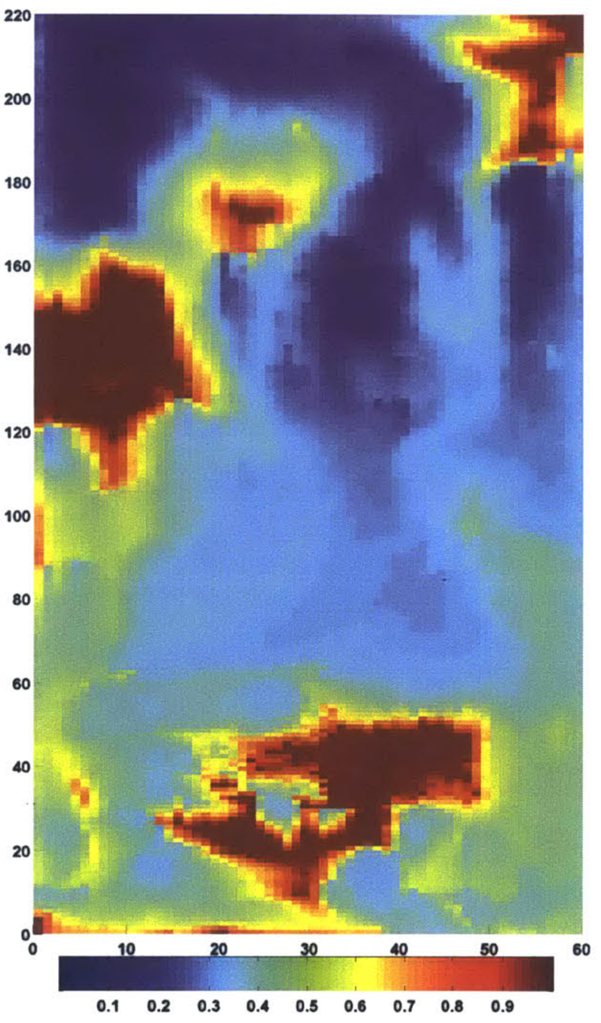
(a) Final filter form the best MCS run **(b) Final filter form the best MCS-MADS run**

Figure 4.39: Final filter for optimal well placement in Stata model (run 1). Red circles represent the non-dominated infeasible solutions that define the filter. The final filter from MCS-MADS (right) is closer to the feasibility axis.

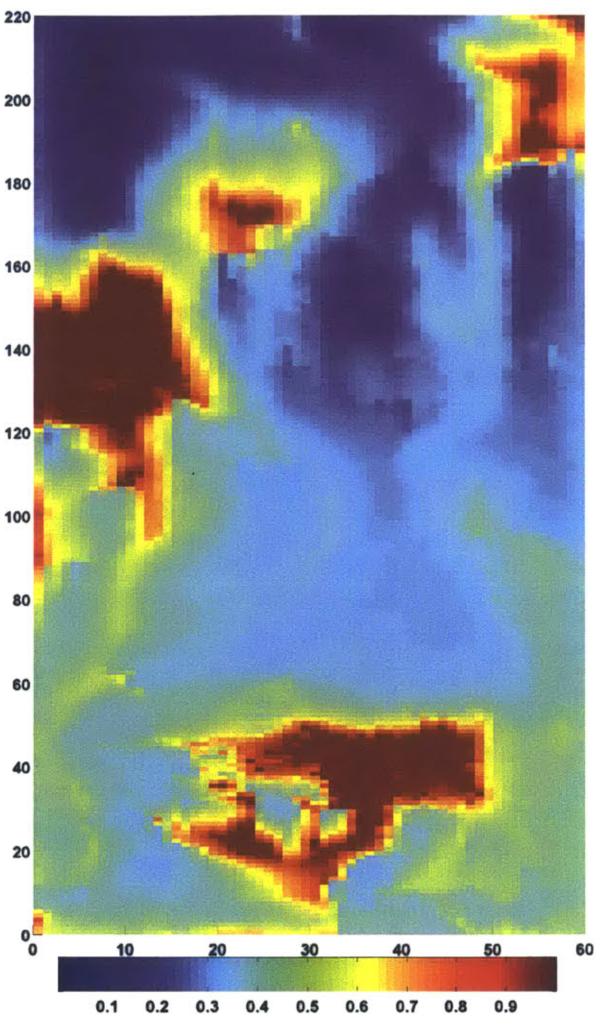


(a) Final filter form the best MCS run (b) Final filter form the best MCS-MADS run

Figure 4.40: Optimal well locations displayed on final oil saturation map of layer one of Stata model for Case 2 (red indicates oil while blue is water). MCS-MADS algorithm results in better well placements that displace larger volumes of oil.

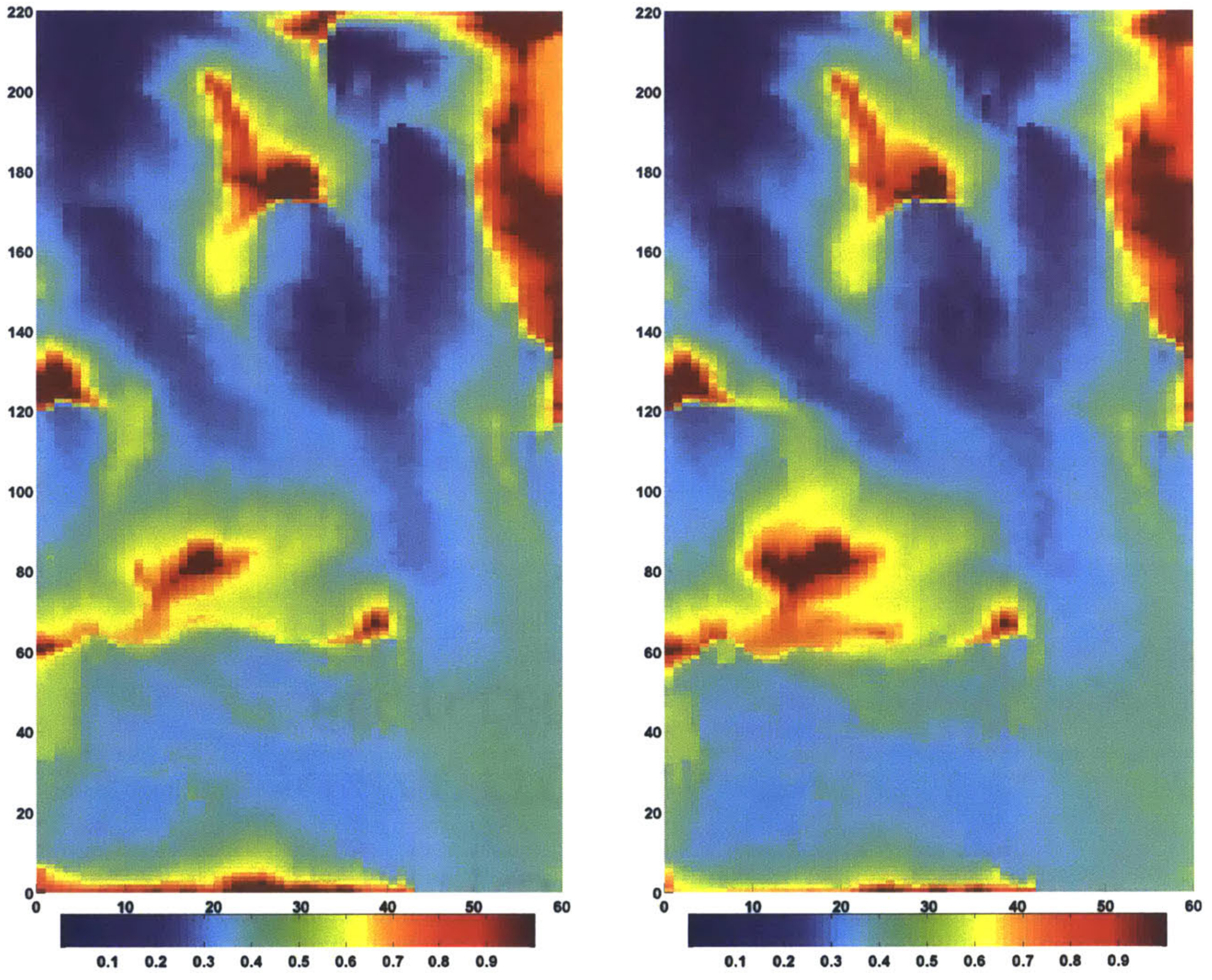


(a) Final filter form the best MCS run



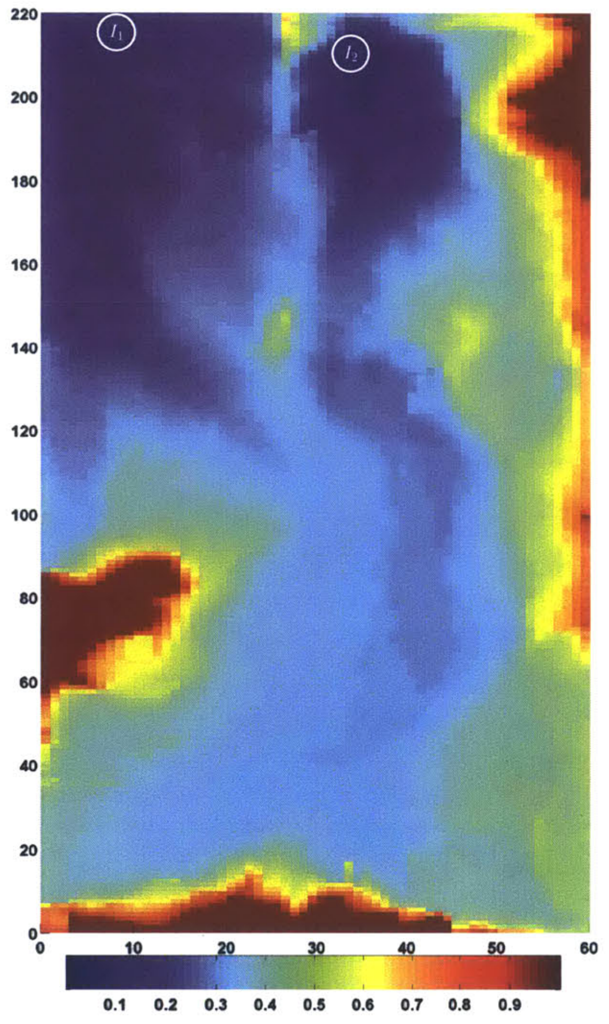
(b) Final filter form the best MCS-MADS run

Figure 4.41: Optimal well locations displayed on final oil saturation map of layer two of Stata model for Case 2 (red indicates oil while blue is water).

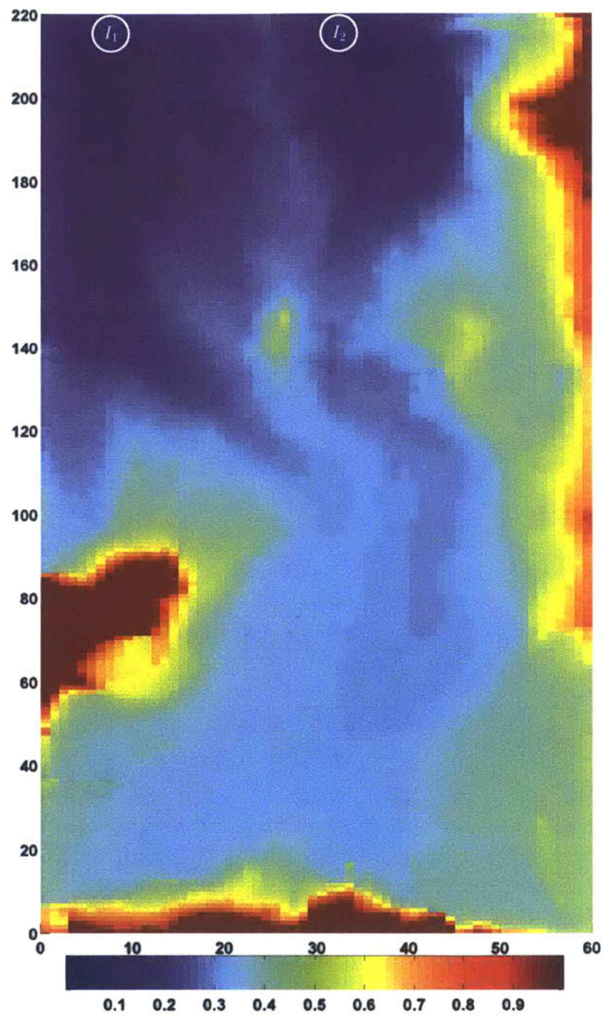


(a) Final filter form the best MCS run (b) Final filter form the best MCS-MADS run

Figure 4.42: Optimal well locations displayed on final oil saturation map of layer three of Stata model for Case 2 (red indicates oil while blue is water).

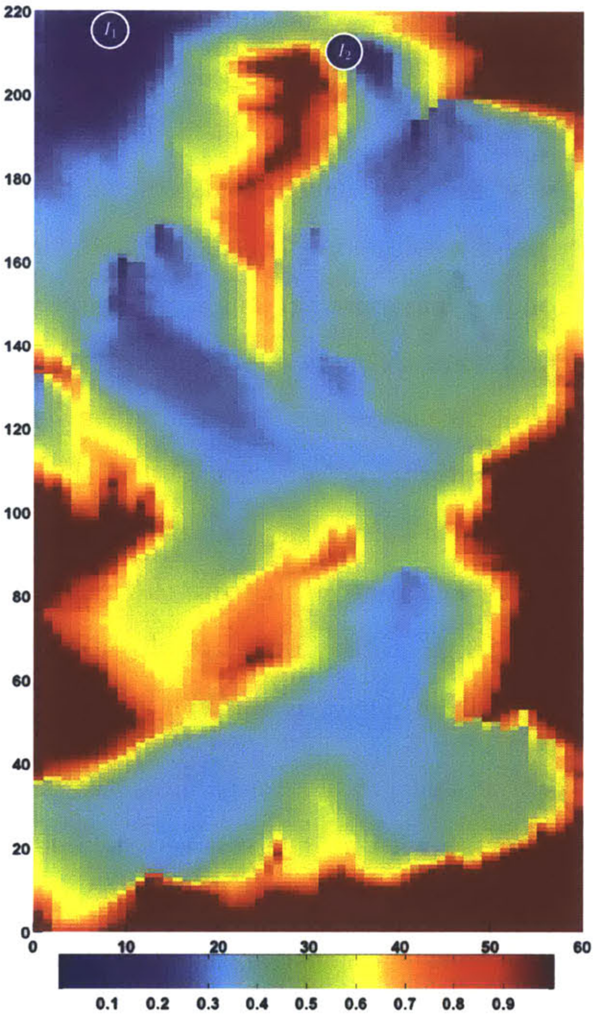


(a) Final filter form the best MCS run

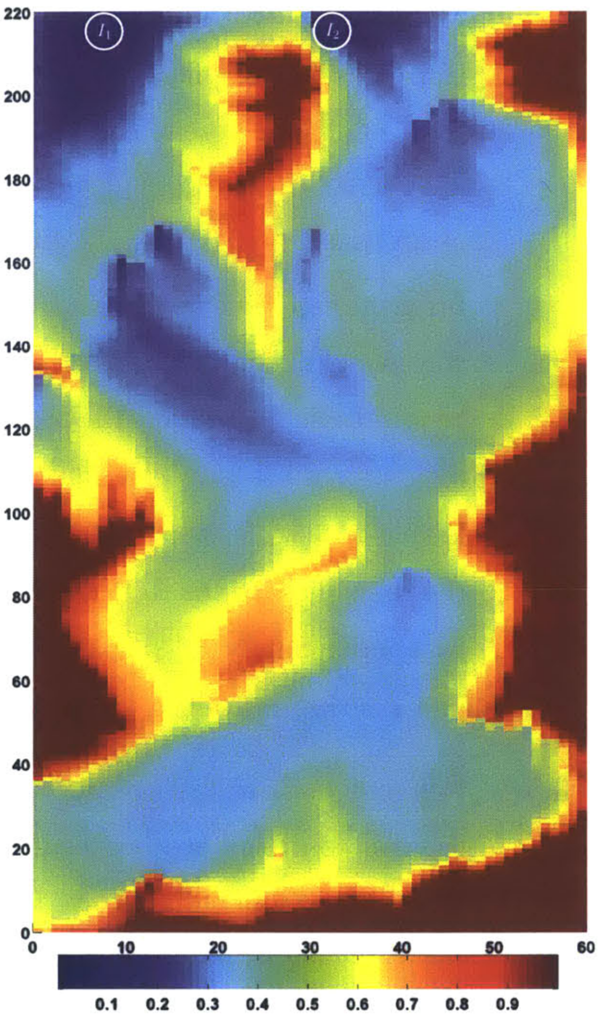


(b) Final filter form the best MCS-MADS run

Figure 4.43: Optimal well locations displayed on final oil saturation map of layer four of Stata model for Case 2 (red indicates oil while blue is water). Both algorithms place injectors at the northern part of the field with small differences due to the local search ability of MCS-MADS algorithm.



(a) Final filter form the best MCS run



(b) Final filter form the best MCS-MADS run

Figure 4.44: Optimal well locations displayed on final oil saturation map of layer five of Stata model for Case 2 (red indicates oil while blue is water). Both algorithms place injectors at the northern part of the field with small differences due to the local search ability of MCS-MADS algorithm.

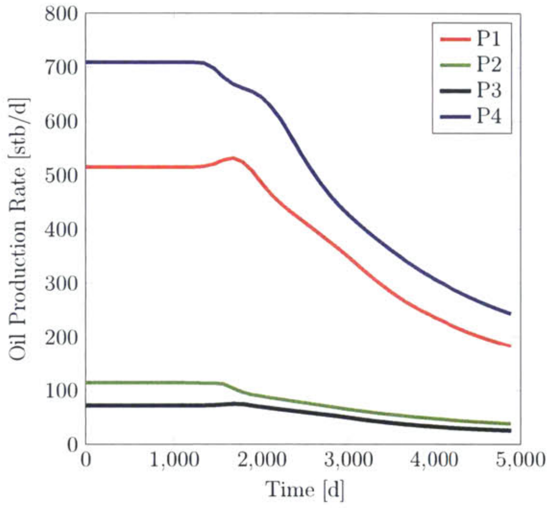
4.2.4 Well Controls Optimization

We now apply optimal well controls using the optimal well placements attained from MCS-MADS. We use NPV as the objective function to optimize well controls in this case to ensure highest economic revenue. The equivalent initial NPV (converted from minimum Lorenz coefficient) is \$220 MM. This case considers incompressible flow which means that injected fluid has to equal produced fluids (oil and water) at all times. Thus, both objective functions physically aim to primarily delay water breakthrough time even though they approach it differently. A single control period is considered in this case (well BHP value is constant throughout the entire production time). Table 4.10 details the progress of the optimization run. A total of twelve trust-region steps are performed (five of which are accepted). We observe that the ROM is able to yield feasible solutions at the end of each trust-region run which illustrates the robustness of the ROM approach. Optimal well controls improve the initial NPV by approximately 11%. This is very significant given that wells are already located in areas that maximize oil recovery and minimize water breakthrough time.

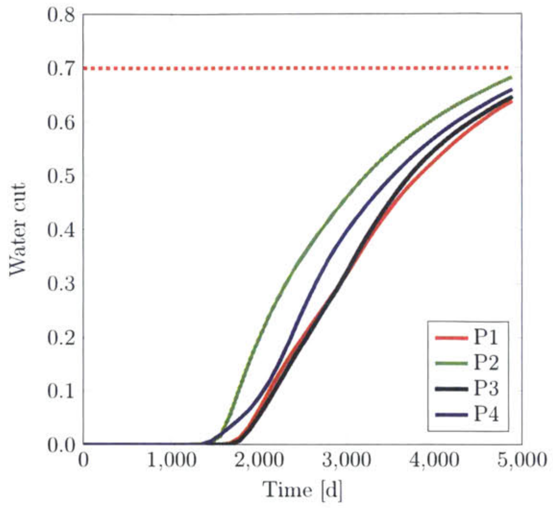
Figure 4.45 displays the optimal BHP controls and production results. We observe that all controls have been adjusted (we started with all controls set at 2000 psi). Water cut constraint is active for well P_2. Well P_4, which is the highest producer, is chocked back to maintain water cut below the assigned constraint.

Table 4.10: Progress of NPV and nonlinear constraint at the end of each trust-region step for ROM and FOM (Stata model)

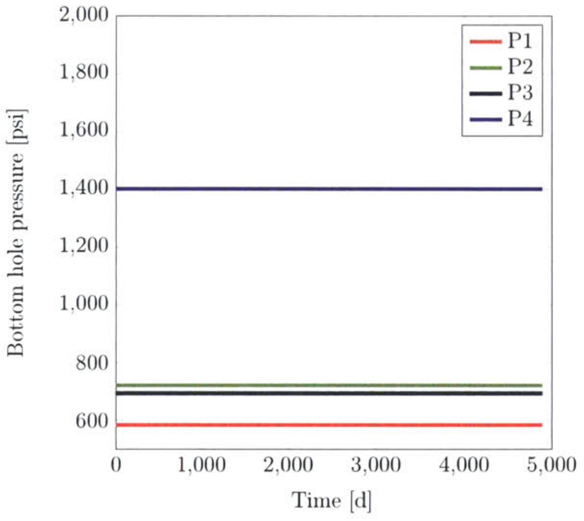
k	ρ	$J_k^{\mathcal{R}}$	$h_k^{\mathcal{R}}$	J_k	h_k
#		[$ MM]		[$ MM]	
1	0.67	222	0	223	0.001
2	0.83	227	0	229	0
3	-0.4	229	0	224	0
4	2.5	237	0	229	0
5	0.23	240	0	242	0
6	2.5	245	0	244	0
7	-1.6	253	0	239	0
8	-0.3	245	0	245	0
9	-0.75	248	0	240	0
10	-0.23	248	0	231	0
11	-0.75	248	0	240	0
12	-4	249	0	243	0



(a) Resulting oil rates.



(b) Resulting water cuts.



(c) Optimal well controls.

Figure 4.45: Optimal BHP controls and resulting production rates and water cut for Stata model optimization study.

4.3 Summary

In this chapter, we applied the developed decision-making framework based on ROMs to two example cases. The first example comprises a sector model of a realistic field located in the Middle East—referred to as Pierce Model—with 25,194 grid blocks whereas the second example—referred to as Stata Model—comprises the top five layers of the SPE 10th comparative study with a total of 66,000 grid blocks. These two examples span different levels of complexities including several vertical wells, multiple horizontal wells, several control periods, and different types of constraints. For the well placement problem, the MCS-MADS hybrid approach utilizing ROMs for local search exhibited superior performance over the standalone MCS in terms of mean objective function, standard deviation, and probability to achieve feasible solutions. This demonstrates the robustness of the MCS-MADS hybrid approach, which combines local and global search capability. While MCS-MADS indeed consumes more simulation runs, it was shown that significant improvement in the objective function value was attained with minimum increase in computational demands due to the use of ROMs which have low computational cost. For the well control problem, the developed trust-region approach based on ROMs exhibited tremendous computational savings while achieving optimal solutions that are very close to those obtained using FOM. Trust-region approach provides the ability to assess and retrain FOMs. This ensures that solutions provided by ROMs are always in close agreement to those obtained using FOM. Although we presented the applications of ROMs for field development, the use of ROMs can be easily extended to other applications such as uncertainty quantification.

Chapter 5

Conclusions and Future Work

5.1 Summary and Conclusions

The main objective of this dissertation is to develop and apply reduced-order models for field development decision-making. It is divided into two components, the reduced-order modeling and the optimization framework for field development strategy. We now summarize each component and present recommendations for future work.

- A new reduced-order modeling approach was developed to reduce computational demands and enable practical applications of reservoir management decision-making. The approach incorporated the use of a new model reduction technique based on Discrete Empirical Interpolation method (DEIM), Proper Orthogonal Decomposition (POD) and Truncated Balanced Realizations (TBR) to build a reduced version of the full-order model that was much cheaper to evaluate, yet accurately reproduced the full models input/output behavior. The latter two techniques allowed for concise representation of the FOM in terms of a relatively small number of variables while the former technique enabled efficient treatment of nonlinear terms in the low-order space. POD procedure constructs reduced basis by running FOM several times and recording solution snapshots. These snapshots dictates the accuracy of the ROM and hence snapshots should contain key flow features. TBR on the other hand performs a change of coordinates where states are order from most important in terms of preserving flow characteristics to least important. Since the computational requirements to construct basis via TBR is substantial, a two-stage reduction procedure was devised where POD was first used for intermediate model reduction before TBR was used to construct the final reduced basis. This two-stage reduction was mainly applied to the pressure equation to reduce the solution pressure states. The saturation equation contained nonlinear terms and therefore required special treatments. In addition to using POD basis to reduce the saturation states, DEIM was used to efficiently reconstruct nonlinear terms and ensure that there was no dependence on the original full dimension. This was accomplished by collecting nonlinear snapshots and applying SVD to construct a new set of orthogonal basis. The DEIM algorithm was then applied to select interpolation indices (gird blocks) that were capable of preserving the continuity of the saturation equation while performing nonlinear evaluations at only subset of grid blocks. We extended the formulation of DEIM in this thesis to include three-dimensional reservoir models with gravitational effects.

- The decision-making component included proposing optimal well locations and controls via the use of optimization methods that enhance the overall field recovery. The core optimization framework for well placements included a local search method (MADS), a global search method (MCS), and a new MCS-MADS hybrid that combined the positive aspects of the previously standalone methods. A major contribution here was the extension of the hybrid approach to take advantage of the devised ROM procedure for well placement optimization. For well control optimization, adjoint procedures were used to efficiently compute gradients that were used within a trust-region framework. Reservoir management is often performed under constraints which are usually nonlinear. These constraints were treated via a filter method. The filter method is perfectly suitable as it does not require any parameter tuning, in addition to its ability to provide useful information regarding trade-off between objective function and constraints. The filter method had been previously incorporated in particle swarm optimization (PSO) and MADS. We extended this procedure to MCS algorithm.

ROMs are sufficiently accurate within a restricted zone around the root-point (the decision variables used to construct the ROM) and thus they need to be updated in a systematic manner over the course of the optimization. A major contribution of this work was the development of adaptive framework that allowed for the use of ROM to reduce computational demands. The framework also included error assessment procedure to ensure minimum deviation from the FOM. During well placement optimization, ROMs were used within the MADS algorithm as an acceleration proxy to locally search for optimal solutions. A calibration parameter was computed at the end of each MADS iteration to ensure the accuracy of the ROM. We note here that we used FOM for global search while the ROM was used for local search without interference. That is, the exchange of solutions between MCS (the global optimizer) and MADS (the local optimizer) was performed using FOM to ensure solution consistency. In well control optimization, gradient optimization was coupled with trust-region method to achieve optimal solutions. The trust-region method was a natural choice for ROM-based optimization as it restricted the optimization step within the

ROM's validity, in addition to synchronizing ROM updates with information obtained during the optimization.

The general decision-making framework, including ROM construction, was applied to find optimal well placements and controls for two reservoir models (one is a realistic model). The optimization problem included bound constraints to ensure wells remained within the boundaries of the reservoir as well as controls to be within the operational limits. In addition, nonlinear constraints were imposed such as minimum well-to-well distance, minimum oil production rate, maximum liquid production rate, and water cut to satisfy any surface plant limits. The key findings from these example cases are as follows:

- MCS-MADS was shown to provide better solutions than the standalone MCS for most cases. This proved that the inclusion of local search procedures can improve the quality of the overall approach by either feeding the global search optimizer with better solutions or by further improving the quality of the final global solution.

- The filter method used within MCS, MCS-MADS, and trust-regions proved to be competent constraint handling techniques with vast generality. It was able to find feasible solutions with modest number of function evaluations even when starting from infeasible solutions. In addition, the filter method provided the capability to perform sensitivity analysis at no additional cost.

- ROMs were shown to be able to reproduce the FOM input/output behavior with high level of accuracy for both objective function and constraint violation. The developed ROMs provided substantial computational speedups of O(100) when tested on three-dimensional models with gravitational forces.

- The incorporation of ROMs in the hybrid MCS-MADS approach proved to improve the optimal solution quality while incurring minimal computational costs. For the Pierce model, the mean NPV improved by 22% while only increasing the computational cost by only 3%. This exhibits the robustness of our general framework which efficiently combines FOM and ROM into the well placement problem.

5.2 Recommendations for Future Work

- Our reduced-order modeling procedure was applied to incompressible oil-water system. This allowed us to apply POD-TBR on the linear pressure equation and POD-DEIM on the nonlinear saturation equation. We recommend extending the ROM approach to compressible systems with multiphase flow. This will require applying DEIM to both pressure and saturation equations.
- Computing the Grammians to obtain TBR transformation matrices requires solving two Lyapunov equations. For large-scale models, the pressure system matrix may have singular values that are widely spread (up to eight order of magnitude difference) which may result in ill-conditioning Lyapunov equations. We thus recommend investigating the use of approximate balanced reduction schemes such as the smith method [71], and the alternate direction iteration (ADI) method [91].
- During development of reduced-order models, we construct a single orthogonal basis for each state and nonlinear term that encodes important flow characteristics. Thus, larger orthogonal basis matrix are required to cover complex flow behavior such as in compositional simulation. Future work may include developing multiple orthogonal bases of smaller size to encode complex flow

behavior. An interesting starting point is presented in Appendix B where localized DEIM (LDEIM) is applied to construct multiple orthogonal bases where each can target specific flow behavior [110]. Initial results shows improvement in solution accuracy as well as runtime speedup.

- Although we presented the concept of relative omitted energy in Sectoin 2.2 as a criterion to determine the number of retained orthogonal basis, the orthogonal basis determination process remains to be heuristic. Work presented in [52] has shown that retaining basis with small energy—though increases solution accuracy when similar training controls are used—may results in unstable ROMs when applying ROMs in optimization. Comprehensive stability analysis may lead to developing more sophisticated selection criterion that preserve ROM stability.

- Our reservoir management optimization approach applied a sequential procedure where we solve the well placement problem assuming a set of well controls before solving the well control problem. This is suboptimal in the sense that there is a strong dependency between well placements and controls. We suggest applying a joint procedure, similar to that in [61], where we jointly optimize for well placements and controls at the same time. This joint optimization will of course require developing a new framework for ROM update and accuracy assessment.

- The use of other optimization algorithms should be considered including other varieties of CS families and firefly algorithms. A particularly interesting scheme is to hybrid MCS and PSO to develop a more robust optimization algorithm.

- ROMs were used in this work for optimization and decision-making. Another venue that is suitable for ROM is uncertainty quantification. The developed

ROM should be coupled with an uncertainty quantification scheme such as Markov Chain Monte Carlo (MCMC) to quantify geologic uncertainties.

- The optimization framework has thus far involved optimizing only a single objective function. A possible extension is to improve the framework to handle multi-objective problems.

- The example cases considered in this work involved developing fields under waterflooding. Other recovery processes such as CO_2 EOR and gas injection are of particular interest. In addition, most techniques developed in this work are general and we therefore recommend extending their applications beyond the field of petroleum engineering.

Appendix A

Recovery Enhancement Using MCS

A.1 Introduction

In this section, we compare MCS to one of the most popular metaheuristic optimization methods, genetic algorithm (GA). Two example cases involving two-dimensional synthetic reservoir models are presented. The first case compares the performance of MCS to that of genetic algorithm (GA) to maximize oil recovery by optimizing the location of four injection wells. The second case entails the use of filter-based MCS to maximize NPV under maximum water cut constraint at the production well.

A.2 Example cases

The described filter-based MCS will now be applied to a synthetic field development optimization problem subject to waterflooding. Two example cases are presented. The first example will depict a comparison between one of the well known meta-heuristic optimization methods—genetic algorithm— and MCS for the well placement problem. The second example will use MCS to maximize net present value in the presence of nonlinear constraints. The geologic model, also used in MRST example cases, represents a sector of the first layer of the SPE 10th comparative study [34]. The grid is $60 \times 60 \times 1$ ($N_x \times N_y \times N_z$, where N_k is the number of grid blocks in direction k). The permeability field is shown in Figure A.1, together with the initial well locations. There are four injection wells located at the corners of the model and one production well located at the middle. Table A.1 represents key simulation and optimization parameters.

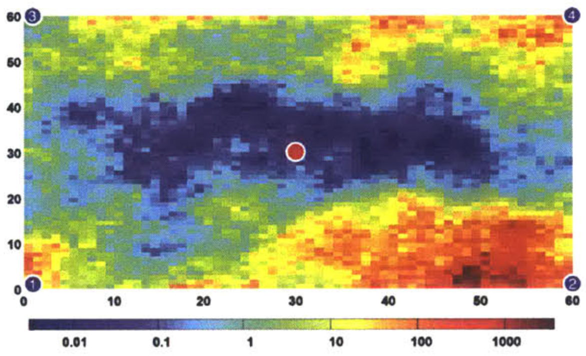


Figure A.1: Permeability in the x-direction in mD with initial well locations, blue circles are injection wells and red circle represents the production well

The system is incompressible and capillary pressure and gravity effects are neglected. The relative permeabilities for the oil and water phases, with initial water and residual oil saturations set to zero, are specified as:

Table A.1: Simulation and optimization parameters

Physical cell dimensions	20 ft $\times$ 10 ft $\times$ 2 ft
Initial pressure, p_i	3500 psi
Mean areal permeability ($k_x = k_y$)	38.61 mD
Mean porosity, ϕ	0.16
ρ_o and ρ_w	45 and 60 lb/ft^3
μ_o and μ_w	5 and 1 cp
r_o	\$100/STB
c_{wp}	\$10/STB
c_{wi}	\$5/STB
maximum well water cut	0.75
Injection rate range	[0 - 45.14] STB/day
Production BHP	500 psi

$$k_{ro}(S_w) = k_{ro}^0 \left(\frac{1 - S_w - S_{or}}{1 - S_{wr} - S_{or}} \right)^a, \tag{A.1}$$

$$k_{rw}(S_w) = k_{rw}^0 \left(\frac{S_w - S_{or}}{1 - S_{wr} - S_{or}} \right)^b. \tag{A.2}$$

where k_{ro}^0 and k_{rw}^0 are the endpoint relative permeabilities. Here we set $k_{ro}^0 = k_{rw}^0 = 1$ and $a = b = 2$.

In all example cases, the production time frame is 10 years. In case 1, the injection rate for each well is equally set to 45.14 STB/day (equivalent to 3 PVI of total injected water). This case is incompressible which implies that the total injection rate is equal to the production rate. The production rate is kept constant throughout the production time frame as we aim to optimize the well placements in this case. In case 2, the optimal well locations are used to maximize NPV under water cut constraint. The injection rates in this case are manipulated insofar as they maximize the NPV while maintaining water cut at a maximum of 0.75.

A.2.1 Case 1 - Optimal injection well locations

In this example, we seek to find the optimal locations of four injection wells that maximize the oil recovered by the production well. This implies optimizing a total of eight discrete areal (x, y) well locations. To do so, we choose to minimize the Lorenz coefficient defined as:

$$L_c = 2\left(\int_0^1 F d\Phi - 0.5 \right). \tag{A.3}$$

where both the flow capacity F and the storage capacity Φ are calculated using a numerically computed flux field and streamline time-of-flight (TOF). Lorenz coefficient measures the deviation of the $F - \Phi$ diagram from an idealized piston-like displacement. It varies between 0 (homogeneous displacement) to 1 (infinitely heterogeneous displacement) and therefore serves as a unique measure of the flow— or dynamic— heterogeneity. It has been shown by [63] that minimizing L_c results in an optimal volumetric sweep.

The well placement optimization study is performed using MCS and GA to assess the performance of MCS. For MCS, the number of nests is set to 50 nests running for 20 generations. The Lèvy flight step size is set to $A/\sqrt{G}$, where A is the distance between the minimum and maximum optimization variables and G is the generation number. For GA, the population size is set to 50 running for 20 generations. The crossover fraction is 0.8 and the mutation rate is 0.1. There is one elite chromosome that survives each generation. Due to the stochastic nature of both methods, each method is run five times starting with different seeds to minimize statistical randomness.

The optimization results for this case are summarized in Table A.2. The optimization run is repeated five times for each method, that is a total of ten optimization runs.

Table A.2: Final Lorenz coefficient from five runs using MCS and GA (best values are in bold)

Run #	MCS L_c coeff.	GA L_c coeff.
1	0.1079	0.1101
2	0.1039	**0.1085**
3	0.1143	0.1199
4	**0.0996**	0.1205
5	0.0996	0.1272
$< L_c >$	0.1124	0.1172
σ	0.0062	0.0078

The minimum Lorenz coefficient for both methods is highlighted in bold. The last two rows indicate the average $< L_c >$, and the standard deviation. In terms of $< L_c >$, we see that MCS outperforms GA. It also yields the best value in terms of objection function. In fact, MCS achieves two equally good objective function values with different solutions (well locations), which is expected as the objective function surface is very complex with multiple local optima.

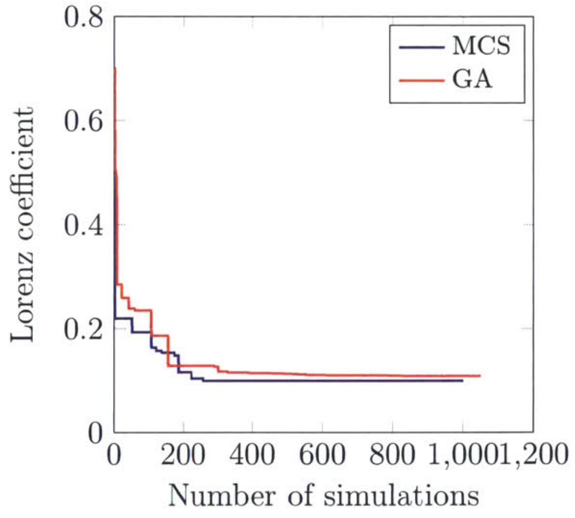


Figure A.2: Evolution of Lorenz coefficient for the best run (case 1)

Figure A.2 depicts the performance of MCS and GA for the best runs (run 4 for

MCS and run 2 for GA). It is clear that MCS exhibits faster convergence rate as it converged after 250 simulation runs only while GA converged after 850 simulation runs. This is due to MCS's robust search strategy which focuses on exploration at early search stages and exploitation at later search stages.

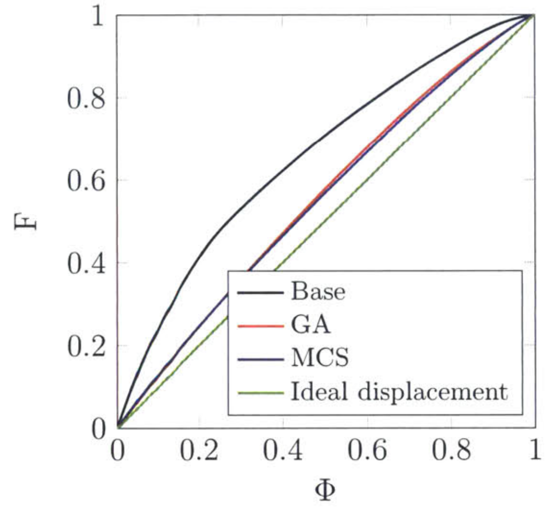


Figure A.3: Storage and flow capacity for base and optimized well locations (case 1)

Storage and flow capacity curves are depicted in Figure A.3. Displacement from MCS optimal well locations exhibits closer behavior to ideal displacement. This behavior is translated into a higher recovery factor (close to 80%) as shown in Figure A.4—an additional increase of 15% over the intuitive base case. Figure A.5 displays the simulation results and well locations for the base solution, and the optimal solutions found using MCS and GA. More oil is evidently extracted from the central field region using optimal well locations suggested by MCS and GA with clearly higher oil volumes extracted using well locations suggested by MCS.

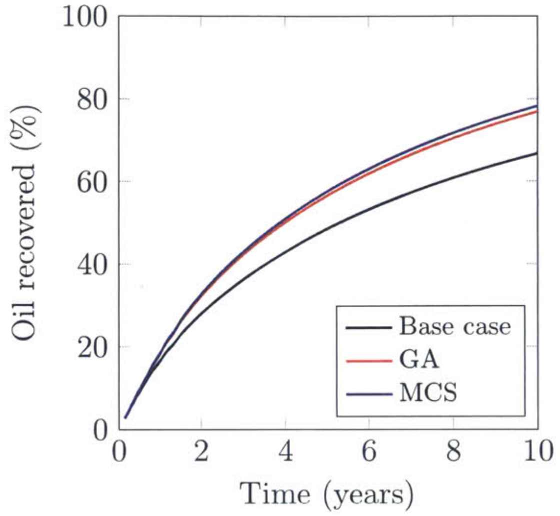


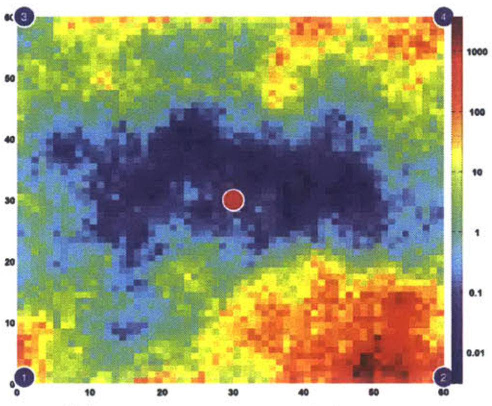
Figure A.4: Recovery factor from base case and optimized well locations (case 1)

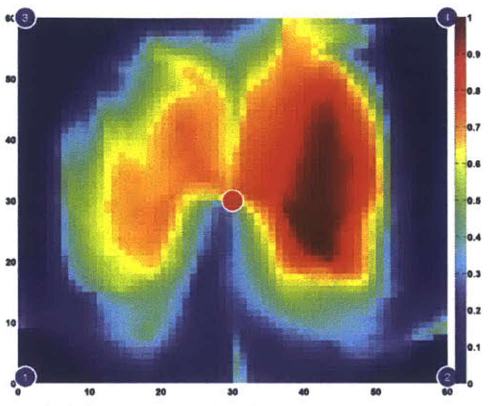
A.2.2 Case 2 - Optimal injection rate with nonlinear constraint

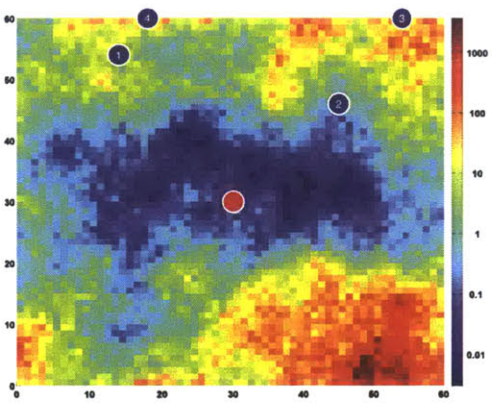
Case 1 above dealt with only bound constraints (well locations bounded by the size of the model). This case will additionally include a nonlinear constraint. Here, we will use the optimal well locations proposed by MCS to maximize the undiscounted NPV by adjusting the injection rates while maintaining a maximum water cut of 0.75 at the production well. Specifically, we seek to maximize the objective function $J(\mathbf{u})$, where

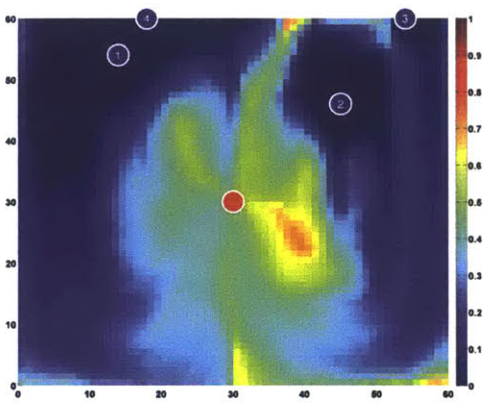
$$J(\mathbf{u}) = r_o Q_o(\mathbf{u}) - c_{wp} Q_{wp}(\mathbf{u}) - c_{wi} Q_{wi}(\mathbf{u}). \tag{A.4}$$

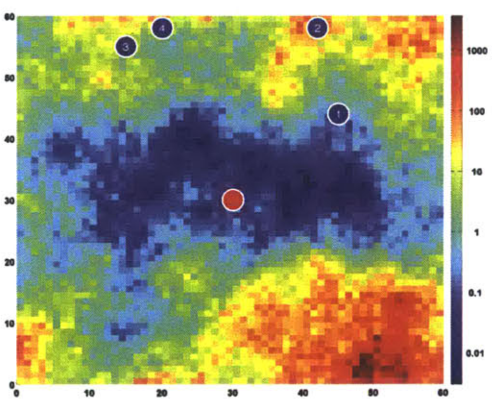
Here r_o is the price of oil (\$/STB), c_{wp} and c_{wi} are the cost of handling produced water and the cost of water injection (\$/STB) respectively. Q_o, Q_{wp} and Q_{wi} are the cumulative oil production, water production and water injection in (STB) respectively. The injection rates are perturbed every 2.5 years comprising a total of 16 control variables. The imposed water cut constraint is nonlinear in nature because the


(a) Base case permeability map


(b) Base case final saturation map


(c) GA-optimized permeability map


(d) GA-optimized final saturation map


(e) MCS-optimized permeability map

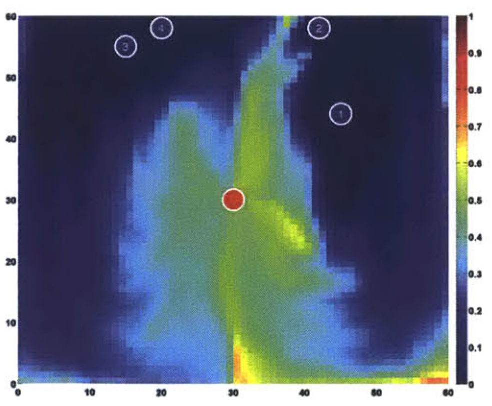

(f) MCS-optimized final saturation map

Figure A.5: Simulation results from the base solution and the MCS and GA optimized solutions, showing the optimal well locations and final oil saturation map

relationship between the control variables (injection rates) and the constraint (water cut) involves reservoir simulation (i.e., nonlinear function evaluations). We use the filter-based MCS described previously to maximize the NPV under constraint with the same parameters used in case 1.

Table A.3: Final NPV from five runs

Run #	NPV ($)
1	**2,093,992**
2	2,084,919
3	2,088,644
4	2,087,366
5	2,089,668
$< NPV >$	2,088,918
σ	3,346

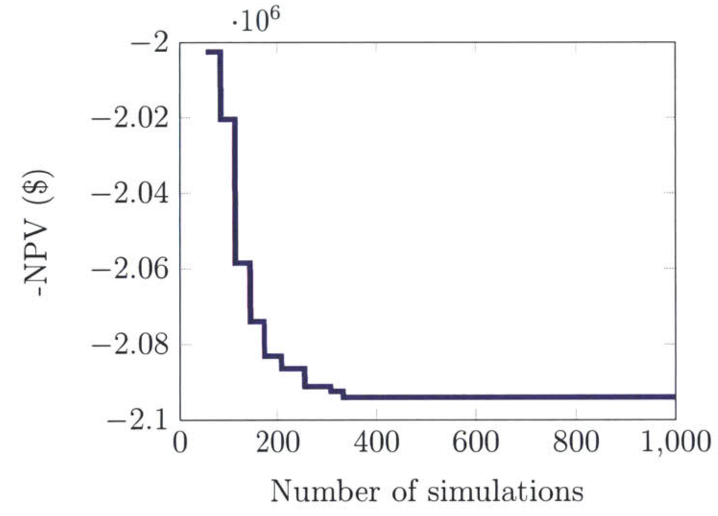


Figure A.6: Evolution of NPV for run 1 (case 2)

Figure A.6 displays the evolution of NPV for the best run (run 1)—the results for all runs are summarized in Table A.3. The stair-line depicting the NPV appears after approximately 50 simulation runs (one generation). This implies that previous simulations lead to infeasible solutions (i.e., solutions that violate the nonlinear constraint).

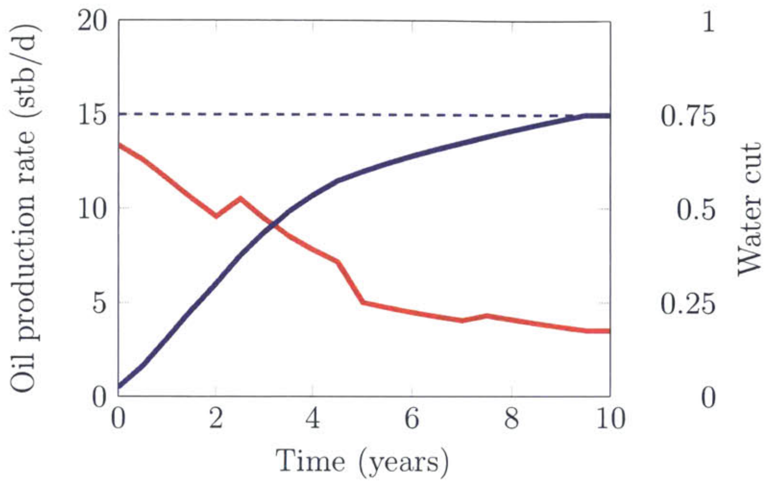


Figure A.7: Oil production rate and water cut (case 2)

Even though all solutions in the first generation were infeasible, the filter-based MCS was able to find feasible solutions at the second generation. Thanks to the initial bias toward exploration, more Lèvy flight searches have been performed and accordingly feasible solutions were quickly achieved with an overall fast convergence toward the optimal feasible NPV. Figure A.7 shows the water cut from the production well with the constraint being active at the final simulation times.

The final filter for run 1 is depicted in Figure A.8. The points in the plot represent the infeasible non-dominated points that define the filter. This filter measures the sensitivity of the objective function $(\mathbf{u})$ to the constraint violation $h(\mathbf{u})$ (similar analogy to the pareto frontier in multi-objective optimization). It quantifies how much the constraint needs to be relaxed in order to achieve more gain in the objective function. For example, if we relaxed the constraint violation by 6.5%—equivalent to changing the maximum water cut from 0.75 to 0.8—we would have been able to achieve a NPV of 2.18×10^6 (an additional 4%). This is due to the obvious reason that higher water cut is associated with more oil production sold at \$100/STB.

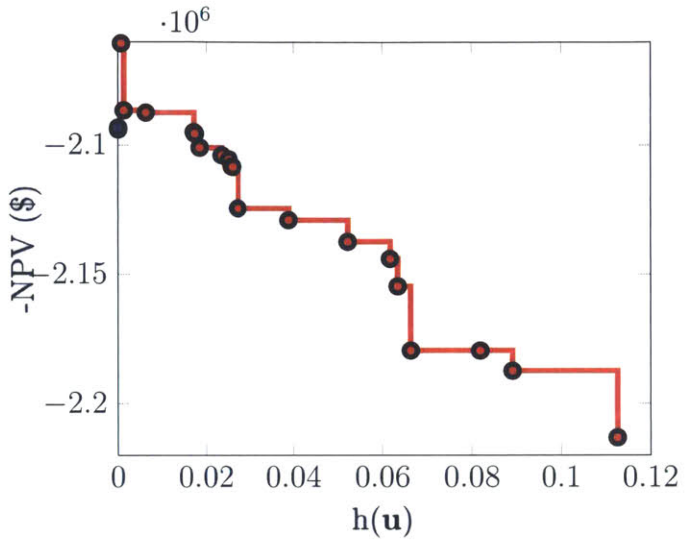


Figure A.8: Final filter for run 1 (case 2)

A.3 Conclusions

we present modified cuckoo search (MCS)—a new metaheuristic global optimization method—for petroleum applications. The method is used to optimize well placement and injection controls. We couple MCS with filtering technique for nonlinear constraint handling. With this approach, the problem is viewed as a bi-objective optimization in which we seek to minimize the constraint violation first before optimizing the objective function. Two example cases have are presented. The first example involves optimizing the location of four injection wells to maximize recovery. The optimization is carried out using GA and MCS to provide a basis for performance. MCS is able to outperform GA in terms of better results and rate of convergence (although the results are less pronounced for this small example). The second example involves optimizing the injection controls to maximize net present value (NPV) under maximum water cut constraint (nonlinear constraint). A filter-based MCS is used to guide the solution. The filter-based MCS is efficient in terms of finding a feasible solutions due to the initial increase in the number Lèvy flight search. The filtering technique provides a sensitivity measure between the objective function and the constraint violation which can be used to assess the additional gain in objective function as a result of constraint relaxation. In future work, we plan to rigorously compare MCS with other metaheuristic techniques for petroleum applications. In addition, we plan to incorporate the use of reduced-order models with MCS to optimize well placements and controls which can address computational challenges associated with reservoir decision-making.

Appendix B

Development of Reduced-order Reservoir Models Using Localized DEIM

B.1 Abstract

For applications of numerical reservoir simulation such as production optimization or uncertainty assessment, hundreds or even thousands of simulation runs are required. Such a huge computational cost is a major constraint in petroleum engineering applications, and hence reduced-order model (ROM) has been intensively studied as an alternative to overcome the high computation cost. While POD-based reduced-order models are generally associated with lower computational costs compared to full order models (FOM), it is not quite efficient when it is applied to a typical large dimensional nonlinear reservoir system. One of the reasons for the inefficiency is that in a typical nonlinear reservoir system, POD-based ROM still depends on the dimension

of the FOM. This is due to the fact that to compute the reduced nonlinear term in the mass balance equation, one must first reconstruct the full-order state solution, and then evaluate the full-order nonlinear term before projecting it onto a reduced subspace. To free the projection process from the full-problem dimension, we develop a reduced-order reservoir model that is based on a discrete empirical interpolation method (DEIM) to approximate nonlinear potential terms so that the repeated online evaluations of the ROM in Newton iteration are independent of the full-order dimension. The independence comes from the fact that DEIM needs to evaluate the nonlinear terms only at interpolation indices that represent grid blocks that are important in terms of preserving the continuity properties of the mass balance equation. A case study is carried out to investigate the performance of DEIM compared to POD. Although the testing schedule of well controls is far apart from the training schedule of well controls, close matches are obtained. Thus, the ROM using DEIM is expected to enable the practical application of reservoir simulator, such as production optimization in which many simulation runs must be performed, in a reasonable time frame by significantly relieving the required numerical effort.

B.2 Introduction

One of the primary goals of petroleum reservoir modeling and management processes is to enable decisions that determine the direction and course of billions of dollars every year. A decision-making framework requires identifying a methodology to propose possible scenarios and develop an efficient technique to evaluate them. This can be mathematically translated into an optimization problem where efficient optimization methods such as gradient-based algorithms are used to propose possible scenarios and reservoir simulation models are used to evaluate them. The use of finite-difference full-physics reservoir simulation models for reservoir optimization and decision-making is computationally expensive. A typical reservoir model might require several hours of run time, and the number of simulation runs required to make a particular reservoir management decision can be in the order of hundreds to even thousands. This high computation cost hinders the popularity of optimization methods in reservoir management decision-making.

Reduced-order models (ROMs) are used as alternatives to full-order models in order to reduce computational demands and enable the practical application of reservoir management decision-making. Reduced-order models generally incorporate the use of model reduction techniques to build a reduced version of the full-order model (FOM) that is much cheaper to evaluate, yet accurately reproduces the full models input/output behavior.

One of the widely used reduced-order modeling techniques is proper Orthogonal Decomposition (POD). It constructs reduced basis by running high-order simulation models several times using different set of forcing input controls, i.e.; production flowrates that covers a particular range of forcing input controls. These runs are called *"training runs"* and the accuracy of the ROM is dictated by how well the training runs cover the solution space. POD has been applied in various applications, i.e.; [27, 113, 70], including reservoir simulation, [29, 30, 54, 93, 95], due to its ability to handle nonlinear systems.

Since reservoir simulation equations are nonlinear, reduction techniques, although reduce dimensions in the sense that far fewer variables are present, still depend on the dimension of the original full-order model through nonlinear terms. The complexity of evaluating the nonlinear terms remains dependent on the full-order model hindering the performance and speed of the constructed ROM. This limitation was seen in reservoir simulation by [29], where POD was effective at the linear solver level while full residual and Jacobian equations were still constructed at every iteration of every time step before being reduced. Detailed description of computational complexity of POD with nonlinear model can be found in [32].

In order to use the above mentioned reduction techniques to construct ROM for reservoir simulation, an efficient treatment for nonlinearities must be devised. One approach, Trajectory Piecewise Linearization (TPWL), uses first-order taylor series expansion to approximate nonlinearities around saved *"closest"* equilibrium states, [84, 29, 54, 96]. Reasonable accuracy and substantial speedups were reported by Cardoso and Durlofsky [30] and He et al. [54] when TPWL was used for reservoir simulation. The approach however requires storing Jacobian matrices for both states and controls, in addition to solution states, which occupies substantial disk space as these matrices can be quite large and difficult to manipulate. In addition, it requires output of partial Jacobians such as flux Jacobian and accumulation Jacobian.

Chaturantabut and Sorensen [32] proposed a discrete empirical interpolation method (DEIM) to deal with nonlinearities. DEIM enables the representation of nonlinear terms through evaluation of the full-order nonlinear terms only at few selected grid blocks (interpolation points). It can be thought of as a clever extension to reduction techniques such as POD to retain nonlinearities at a lower dimensional space. DEIM has been applied in various applications, i.e., miscible viscous fingering in porous media [33], and reservoir simulation [93, 8].

To further improve the accuracy stability of DEIM, Peherstorfer et al. [83] proposed a scheme of localized DEIM (LDEIM). Whereas DEIM projects nonlinear terms onto one global subspace, LDEIM computes several local subspaces, each tailored to a particular region of characteristic system behavior. LDEIM uses machine learning methods to discover these regions via clustering. Then, local DEIM approximations are computed for each cluster. Clustering, in addition local basis computation, is performed offline and hence incurs minimal computational demands. The selection of an optimum approximation is conducted via machine-learning-based classification during online phase. By using the multiple local DEIM approximations, it is possible to reduce the number of interpolation points further, and as a result, reduce the computational costs even further. To the best of our knowledge, LDEIM has not been applied for reservoir engineering applications.

In this section, we construct a new reduced-order modeling procedure that does not require the full-order state variables for constructing the nonlinear terms. This is achieved by combining POD method with the localized discrete empirical interpolation method (LDEIM). We show that the use of LDEIM improves the efficiency of the resulting reduced-order model representation via comparison with its counterpart, the vanilla DEIM.

The section progresses as follow. The modeling procedure for the proposed reduced-order model is first derived. Then, it is compared with POD-DEIM-based ROM using a synthetic geologic model. We finally conclude with future plans for this work.

B.3 Reservoir Modeling Procedure

In this section, we describe the governing equations—including the formulation of the problem, the discretization procedure, the solution strategy, and the reduced-order modeling techniques for three-dimensional two-phase flow reservoir model with gravitational forces.

B.3.1 Governing Equations for Two-phase Flow

Reservoir simulation models are derived by combining the constitutive mass balance equation with multiphase Darcy's law. We consider incompressible oil-water flow system and neglect capillary pressure effects. Also, there is no mass transfer between phases, i.e., the oil component resides only in the oil phase while the water component resides only in the water phase. Then, the continuity equation for each phase, designated j (where $j = o$ for oil and w for water), is given by:

$$\frac{\partial}{\partial t}(\phi \rho_j S_j) + \nabla \cdot (\rho_j \mathbf{v}_j) = q_j, \tag{B.1}$$

where ϕ is porosity, ρ_j and S_j are the density and saturation of phase j respectively, $\mathbf{v}_j$ is the phase Darcy velocity, and q_j is the source term. Assuming incompressible flow, i.e., ϕ and ρ_j are constants, The continuity equation (B.1) is simplified into:

$$\phi \frac{\partial S_j}{\partial t} + \nabla \cdot (\mathbf{v}_j) = \frac{q_j}{\rho_j}, \tag{B.2}$$

A more tractable system of equations consisting of a pressure equation and a saturation (fluid-transport) equation can be written for easier dimensionality reduction. The pressure equation (assuming no capillary effects) is given by:

$$\nabla \cdot \mathbf{v}_t = q_t, \qquad \mathbf{v}_t = -\mathbf{K}\Big[\lambda_t \nabla p + (\lambda_w \rho_w + \lambda_o \rho_o) g \nabla z\Big], \tag{B.3}$$

where $\mathbf{v}_t$ is total Darcy velocity, $\mathbf{K}$ is the absolute permeability (assumed to be a diagonal tensor), $\lambda_t = \lambda_w + \lambda_o$ is the total mobility, $\lambda_j = k_{r_j}/\mu_j$ is the mobility of phase j, μ_j is the respective phase viscosity, k_{r_j} is the relative permeability of phase j, p is pressure, $q_t = q_w + q_o$ is the total flow rate, g is the gravity acceleration constant, and ∇z is the negative upward vertical direction. The derived saturation equation is given by:

$$\phi\frac{\partial S_w}{\partial t} + \nabla \cdot \Big(f_w(S_w)\big[\mathbf{v}_t - \lambda_o(\rho_w - \rho_o)g\mathbf{K}\nabla z\big]\Big) = q_w. \tag{B.4}$$

where $f_w(S_w) = \lambda_w/(\lambda_w + \lambda_o)$ is the water fractional flow. The term $f_w(S_w)\mathbf{v}_t$ represents viscous forces while the term $f_w(S_w)\lambda_o(\rho_w - \rho_o)g\mathbf{K}\nabla z$ represents gravitational forces.

B.3.2 Discretization and Solution Strategy

The two-phase flow description for incompressible flow and no capillary effects entails three equations and three unknowns (p, S_o, S_w). We select p and S_w as the primary unknowns. Once these primary unknowns are computed, S_o can be readily determined from the saturation constraint, i.e., $S_w + S_o = 1$. For the sake of brevity, we let S_w $= S$ as the water saturation.

The pressure equation (B.3) and saturation equation (B.4) are nonlinearly coupled through the saturation-dependent mobilities in the pressure equation and through the pressure-dependent total velocity in the saturation equation, in addition to other terms that depend on pressure, e.g, viscosities. Following procedures developed by Lie et al. [69], a sequential method is applied to obtain solution states where saturation from previous step (or initial condition) is used to compute the saturation-dependent coefficients, e.g., λ_t in (B.3), before it is solved for pressure and subsequently total velocity. Then, total velocity is kept constant while saturation from (B.4) is solved and advanced in time. Next, the new saturation states are used to update the saturation-dependent terms in (B.3) and pressure states are solved again, and so on. The pressure equation in (B.3) is discretized explicitly while the saturation equation in (B.4) is discretized implicitly in time as follow:

$$\mathbf{T}^n \mathbf{p}^{n+1} - \boldsymbol{G}^n = \mathbf{B}\mathbf{u}^{n+1}, \tag{B.5}$$

Here $\mathbf{T}^n \equiv \mathbf{T}(S^n)$ is a diagonal transmissibility matrix that relates flow in phase j to the difference in pressure. It is calculated using a two-point flux approximation scheme (see [18] for details). The superscript n represents time step, $\boldsymbol{G}^n \equiv \boldsymbol{G}(S^n)$ is a vector containing the gravitational effects, $\mathbf{u}$ is the input controls (boundary conditions), i.e., well flow rates or BHP, $\mathbf{B}$ is the arrangement matrix for the controls and $\mathbf{p}$ is the unknown pressure vector we seek to solve. The discretized pressure equation (B.5) is a linear equation as the transmissibility matrix $\mathbf{T}^n$ does not depend on pressure (since viscosity is constant for incompressible flow). The saturation equation is descritized using finite volume:

$$\mathbf{S}^{n+1} = \mathbf{S}^n + \frac{\Delta t}{\phi}\Big[\mathbf{F}^{n+1} + \mathbf{Q}_w\Big], \tag{B.6}$$

Here $\mathbf{Q}_w$ denotes the source/sink vector and $\mathbf{F}^{n+1} \equiv \mathbf{F}(S^{n+1})$ is the numerical approximation of the total flux from viscous and gravitational forces across all grid block interfaces, e.g., for a grid block Ω_i and associated interfaces γ_{ij} (where ij represents the shared interface between the grid block Ω_i and the neighboring grid blocks Ω_j):

$$F_i^{n+1} = \int_{\Omega_i}\int_{\gamma_{ij}} f_w(S^{n+1})_{ij}\big[v_{ij} - g_{ij}(S^{n+1})\big]\,\mathrm{d}V, \tag{B.7}$$

$f_w(S^{n+1})_{ij}$ designates the fractional flow function associated with γ_{ij}, v_{ij} is the Darcy flux, and $g_{ij}(S^{n+1})$ is the gravitational flux across the interface. The total flux term (referred to as flux in this work) can be seen to introduce nonlinearity in (B.6) as it is a function of the saturation state we seek to solve. It is also clear from (B.7) that the flux term introduces a direction dependency into the system. It is therefore treated using *"upstream weighting"*. Specifically, the evaluation of the flux term depends on the direction of flow, as follows:

$$f_w(S^{n+1})_{ij} = \begin{cases} f_w(S_i^{n+1}) & \text{if } v_{ij} \geq 0, \\ f_w(S_j^{n+1}) & \text{if } v_{ij} < 0. \end{cases} \tag{B.8}$$

Therefore, the nonlinear flux vector in (B.6) represents a non-componentwise function as the computation of each element depends on other spatially neighboring elements due to upstream weighting. If the flux term is evaluated without being upstream weighted, the numerical solution may display oscillations, overshoots, or undershoots (e.g., saturation less than zero or greater than one), or converge to an incorrect solution.

The source/sink term, right hand side in (B.4), represents wells which are the typical boundary condition in reservoir simulation. Wells are modeled using the following well equation:

$$-(q_t)_i^{n+1} = (\lambda_t)_i^n WI(p_i^{n+1} - p_i^w), \tag{B.9}$$

where $(q_t)_i^{n+1}$ is the total volumetric flow rate from block i into the well (or vise versa) at time $n+1$, p_i^{n+1} is the grid block pressure and p_i^w is the wellbore pressure for well w in grid block i, and WI is the well index. For a vertical well that fully penetrates block i, WI is computed using Peaceman well model [82] :

$$WI = \left[\frac{2\pi k \Delta z}{\ln(r_0/r_w)}\right]_i, \tag{B.10}$$

where r_w is the wellbore radius and $r_0 \approx 0.2\Delta x$. Note that, if the well is operating under bottom hole pressure (BHP) control, this is represented in the simulator by specifying p_i^w in the well equation (2.9).

The descritized pressure equation in (B.5) represents a linear system as all parameters are independent of pressure—transmissibility is a function of saturation in the case of incompressible flow. This system is solved using efficient linear routine, i.e., MATLAB's built-in solver [73] for small models or AGMG [78] for large models. In contrast, the discretized saturation equation (B.6) represents a nonlinear set of algebraic equations as the flux is a functions of saturation. Thus, it is solved using Newton Raphson's method. The residual form of (B.6) is expressed as:

$$\mathbf{g} = \mathbf{S}^{n+1} - \mathbf{S}^{n} - \frac{\Delta t}{\phi}\left[\mathbf{F}^{n+1} + \mathbf{Q}_w\right] = 0, \tag{B.11}$$

The Jacobian matrix is represented as:

$$\mathbf{J} = \mathbf{I} - \frac{\Delta t}{\phi}\left(\frac{\partial \mathbf{F}^{n+1}}{\partial \mathbf{S}^{n+1}}\right). \tag{B.12}$$

B.4 Localized Discrete Empirical Interpolation

Localized DEIM (LDEIM) improves the accuracy and stability of the original DEIM through constructing multiple local nonlinear bases where each basis represents distinctive local features of the nonlinear terms instead of constructing a global nonlinear basis that approximates the overall nonlinear terms. This results in using lower number of interpolation points per Newton iteration as each nonlinear local basis approximates the nonlinear terms for a period of time.

Suppose $\mathbb{F} = \{\mathbf{F}_i\}_{i=1}^{k}$ is the set of nonlinear term snapshots where k is the total number of snapshots.LDEIM clusters similar snapshots that share certain flow behavior and partitions them into $\{\mathbb{F}_1, \cdots, \mathbb{F}_k\}$ of k subsets. Then, a local orthogonal basis $\mathbf{\Psi}_i$ is created for each subset using SVD procedures, in addition to its corresponding interpolation points $\vec{\mathbf{z}}_i$ where $\vec{\mathbf{z}}_i = [z_1, \cdots, z_m]^T \in \mathbb{R}^{m\times 1}$ is a vector containing m interpolation indices that are determined inductively via Algorithm 3. A classifier $c \rightarrow \{1, \cdots, k\}$ is trained in the offline phase to select an optimal local DEIM approximation $(\mathbf{\Psi}_i, \vec{\mathbf{z}}_i)$ with respect to an indicator $\mathbf{v} \in V$, Peherstorfer source. The classifier is trained on the indicators of the nonlinear snapshots. In this work, the indicator is defined as:

$$\mathbf{v} = (\mathbf{\Psi}^g_{\vec{\mathbf{z}}})^{-1}\mathbf{F}_{\vec{\mathbf{z}},i} \qquad \text{where} \quad i = 1, \cdots, k \tag{B.13}$$

where $\mathbf{\Psi}^g_{\vec{\mathbf{z}}}$ is the global basis extracted at the interpolation indices $\vec{\mathbf{z}}$ and $\vec{\mathbf{z}}$ it corresponding interpolation points. In the online phase, a k-nearest neighbor algorithm (KNN) is used to determine the classifier (basis and interpolation indices) that is most suitable to an indicator. The indicator in this work is defined as:

$$\mathbf{v} = (\mathbf{\Psi}^g_{\vec{\mathbf{z}}})^{-1}\mathbf{F}_{\vec{\mathbf{z}}}(\mathbf{\Phi}\mathbf{S}^n_r) \tag{B.14}$$

where $\mathbf{\Phi}$ is the state basis obtained from POD.

B.5 Application Example

A case study is conducted to assess the quality of the order reduction methods in terms of predictability with respect to changes in well controls. The tested geologic model is shown in Figure B.1. It comprises a portion of the SPE 10th comparative study model. The grid is $30 \times 110 \times 4$ ($N_x \times N_y \times N_z$, where N_k is the number of grid blocks in direction k) totaling 13,200 grid blocks. The physical dimension of each grid block is 20 ft $\times$ 10 ft $\times$ 2 ft. There are four production wells located at the corners of the model and one injection well located at the center. All of the wells are under flowrate control. Permeability in the x-direction is depicted in Figure B.1. Permeability is taken to be a diagonal tensor, with $k_x = k_y$. The porosity is constant in this model and set equal to 0.25. The initial water and residual oil saturations are zero. For oil, we set $\rho_o = 45$ lb/ft^3, $\mu_o = 5$ cp; for water, we set $\rho_w = 65$ lb/ft^3, $\mu_w = 1$ cp. The system is incompressible and capillary pressure and gravity effects are neglected. The relative permeabilities for the oil and water phases are specified as:

$$k_{ro}(S_w) = k_{ro}^0 \left(\frac{1 - S_w - S_{or}}{1 - S_{wr} - S_{or}} \right)^a, \tag{B.15}$$

$$k_{rw}(S_w) = k_{rw}^0 \left(\frac{S_w - S_{or}}{1 - S_{wr} - S_{or}} \right)^b. \tag{B.16}$$

where k_{ro}^0 and k_{rw}^0 are the endpoint relative permeabilities. Here we set $k_{ro}^0 = k_{rw}^0 = 1$ and $a = b = 2$.

A training set of well production rates are devised to create a training run for the reduced-order model. We specify the injection well to inject water at a constant rate of 100 STB/day, while the four production wells produce also constantly at 25 STB/day. The training case was simulated for 1500 days using MATLAB Reservoir Simulation Toolbox [69]. Snapshots of saturation and nonlinear phase potential term are collected during every time step. The saturation bases are then constructed using POD with 35 columns. We thus reduced the dimension of the problem from 13,200 variables to only 35 variables. In addition, the nonlinear phase potential term bases

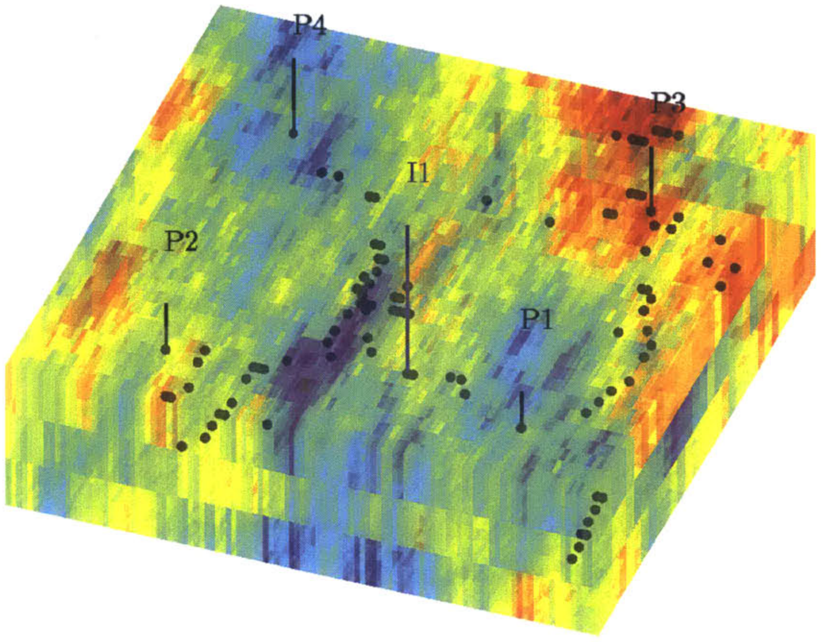


Figure B.1: Portion of the SPE 10 reservoir model (13,200 grid blocks) with four producers and one injector. $\log_{10} k_x$ is displayed with discrete interpolation points in the background.

are constructed with 300 columns. The corresponding 300 interpolations points are selected, which are shown in Figure B.1. For LDEIM, five clusters are considered, and the number of interpolation points in each cluster is $\{149, 98, 72, 183, 246\}$. The average number of interpolation points is 150, which is approximately half the number of the DEIM interpolation points.

To test the performance of the reduced-order models, we follow He [52] which defines a target flowrate schedule for each well. The target flowrate schedule for each well, which is shown in Figure B.2, is randomly generated within an interval between 17 STB/day and 35 STB/day under the constraint that injected fluid (water) has to be equal to produced fluid (oil and water) for incompressible flow model. The target schedule for each well is perturbed every 150 days. It is clear that the training schedule, taken to be constant for all wells, is completely different from the target schedule. We then interpolate between the training and target schedules to enable systematic perturbation away from the training run. In particular, we specify the test case flowrate as follows:

$$\mathbf{u}_{\text{test}} = (1-\alpha)\mathbf{u}_{\text{training}} + \alpha\mathbf{u}_{\text{target}}, \tag{B.17}$$

where $\mathbf{u}_{\text{training}}$ is the training schedule, $\mathbf{u}_{\text{target}}$ is the target schedule, and $\mathbf{u}_{\text{test}}$ is the test schedule interpolated by a weight factor α that is taken to be between 0 and 1. The error between the full-order model and the reduced-order model is expected to increase with increasing α as the test case is entirely the target flowrate schedule when α is 1. The error is quantified by the mismatch of the production rates of fluids. For instance, the oil production rate mismatch is defined as:

$$E_o = \frac{1}{n_{pw}} \sum_{j=1}^{n_{pw}} \frac{\int_0^T |Q^j_{o,\text{FOM}} - Q^j_{o,\text{ROM}}| \quad dt}{\int_0^T Q^j_{o,\text{FOM}} \quad dt} \tag{B.18}$$

where $Q^j_{o,\text{FOM}}$ is the oil production rate from the full-order model, $Q^j_{o,\text{ROM}}$ is the oil production rate from the reduced-order model n_{pw} is the number of production wells, and T is the total simulation time.

Figure B.3 shows the simulation solutions of the oil water production rates and for the four production wells using the training and target rate schedules. The dashed lines show the training simulation solution that will be used to build the ROM, and the solid lines display the reference target case solution, all of which are obtained by using FOM. For the case of $\alpha = 1$, computation times for FOM, POD, DEIM, and LDEIM are presented in Table B.1, in which the accuracy indicator E_O is also presented. As expected, the fastest method is LDEIM. Furthermore, LDEIM outperforms DEIM for

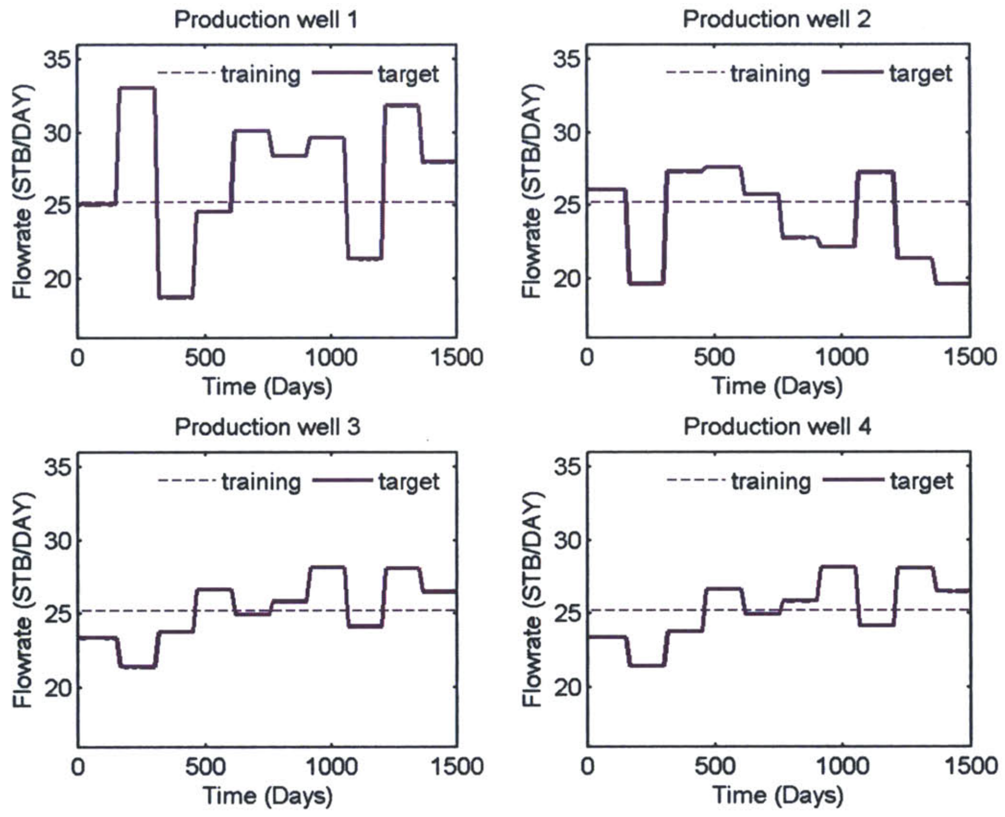


Figure B.2: Training (dashed) and target (solid) well production schedules.

accuracy even with much fewer interpolation points. Figure B.4 shows the simulated flowrates for the four production wells using DEIM. The solid lines are the simulated solution using FOM, while the dashed lines are the results out of DEIM. Although the first production well shows a good match between FOM and DEIM, the other wells show deviation from FOM results especially in the later part of the simulation. This might be because the nonlinear term bases and the corresponding interpolation points are not adequate to represent the physical characteristic for the corresponding period of simulation. On the contrary, the results of LDEIM are in overall agreement with the full-order model simulation results (Figure B.5). This means that the less accuracy in DEIM can be rectified by using the bases and the interpolation points tailored to the corresponding physical behavior. Table B.2 shows the oil production mismatch estimates from using DEIM and LDEIM according to the value of α from 0 to 1. As it is expected, the error increases as α increases, and LDEIM always outperforms DEIM.

Table B.1: Oil production mismatch and computation time of simulation runs ($\alpha = 1$)

	FOM	POD	DEIM	LDEIM
Dimension	13,200	35	L= 50, M= 300	L= 50, $\bar{M} = 150$
E_0 (error)	0	0.010	0.039	0.026
Computation time (min)	563	9	6.9	5.3

Table B.2: Oil production mismatch estimates for reduced-order models (DEIM and LDEIM)

	$\alpha = 0$	$\alpha = 0.2$	$\alpha = 0.4$	$\alpha = 0.6$	$\alpha = 0.8$	$\alpha = 1.0$
DEIM	0.0050	0.0165	0.0240	0.0309	0.0376	0.0436
LDEIM	0.0051	0.0088	0.0133	0.0168	0.0197	0.0212

Overall, the results for all test cases demonstrate that the LDEIM is able to provide simulation results in reasonable agreement with those from the full-order model. For this reservoir model, the full-order simulation runtime is about 563 minutes, while the runtime for the LDEIM is about 5.3 minutes. We thus achieve a reduction of computational time by a factor of 106. Although this reduction in computation is considered modest as compared to computation time reduction of $O(1000)$ attained by Chaturantabut and Sorensen [33] when using DEIM for nonlinear miscible viscous fingering problem, we anticipate larger speedup factors when larger models are used. This is because DEIM eliminates full evaluation of nonlinear terms and therefore less time will be spent during Newton-Raphson iterations which will be very advantageous for large models with large number of variables.

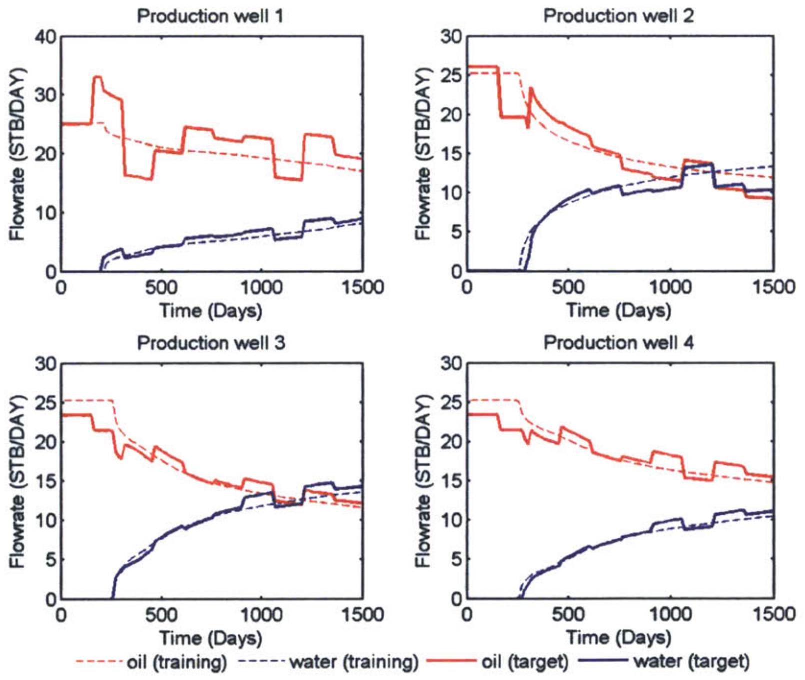


Figure B.3: Training (dashed) and target (solid) simulations production rates.

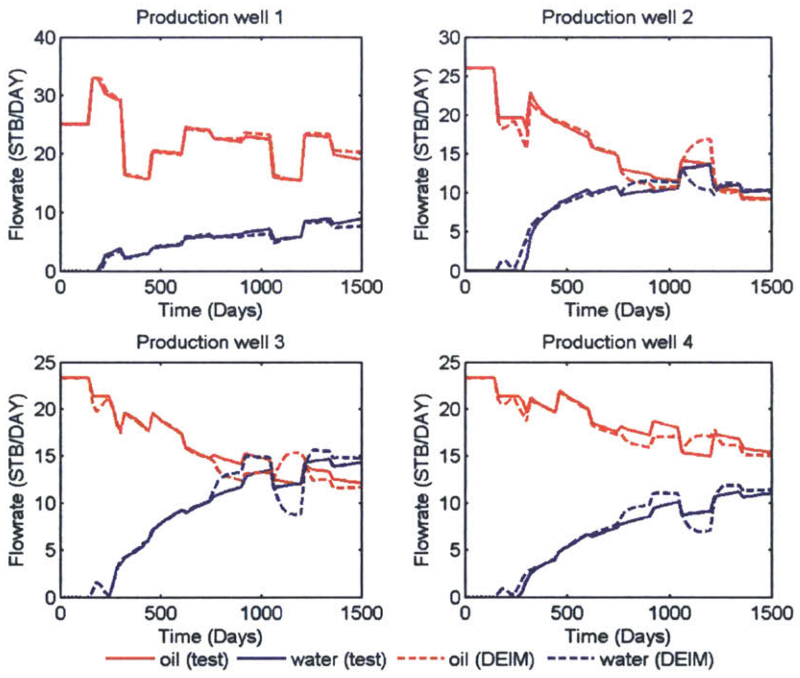


Figure B.4: FOM well production rates (solid) and DEIM production rates (dashed) for $\alpha = 1$.

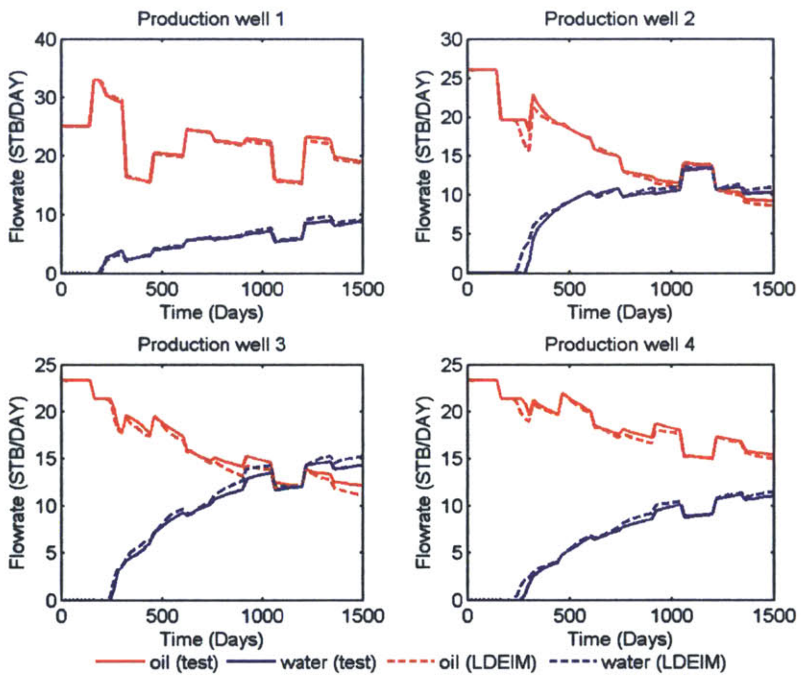


Figure B.5: FOM well production rates (solid) and LDEIM production rates (dashed) for $\alpha = 1$.

B.6 Conclusion

A reduced-order reservoir simulation model based on the combination of proper orthogonal decomposition (POD) and a localized discrete empirical interpolation method (LDEIM) is constructed. The application of LDEIM in 3D-reservoir simulation, in which the nonlinearity is significant due to gravity effects, has not been attempted in the literature. The LDEIM depends on machine learning techniques in order to train localized DEIM approximations and to find a corresponding indicator during an online phase. A case study is carried out using SPE 10 geologic model, and the result supports the use of LDEIM for efficiency with minimal loss of accuracy. Therefore, the proposed reduced-order model based on LDEIM is expected to provide a good tool for efficient reservoir simulation. However, it should be noted that the constructed reduced-order model is evaluated in terms of predictability only with respect to changes in well controls. Thus, the developed reduced-order reservoir simulation model can be utilized for optimizing well control settings, which would require a number of simulation runs when iterative schemes of minimization/maximization are used. In future work, we plan to evaluate the applicability of the developed reduced-order model for determining optimal well settings, also known as production optimization.

Bibliography

[1] Introduction to IPOPT : A tutorial for downloading , installing , and using IPOPT, 2010. URL `https://projects.coin-or.org/Ipopt`.

[2] International Energy Outlook 2014. Technical report, U.S. Energy Information Administration, 2014.

[3] M. Abramson, C. Audet, G. Couture, J. Dennis Jr, and S. Le Digabel. The Nomad Project, 2010. URL `http://www.gerad.ca/nomad`.

[4] A. Agarwal and L. Biegler. A Trust-region Framework for Constrained Optimization using Reduced Order Modeling. *Optimization and Engineering*, **14** (1): 3–35, 2010.

[5] I. Aitokhuehi. *Real-Time Optimization of Smart Wells.* Master's thesis, Department of Petroleum Engineering, Stanford University, 2004.

[6] N. Alexandrov, J. Dennis, R. Michael, and V. Torczon. A Trust Region Framework for Managing the Use of Approximation Models in Optimization. Technical Report 97, Institute for Computer Application in Science and Engineering, 1997.

[7] Z. Alghareeb. *Monitoring and Control of Smart Wells.* Master's thesis, Department of Energy Resources Engineering, Stanford University, 2009.

[8] Z. Alghareeb and J. Williams. Optimum Decision-Making in Reservoir Managment Using Reduced-order Models. Paper SPE 166305-MS presented at the SPE Annual Technical Conference & Exhibition, New Orleans, Lousiana, USA, 2013.

[9] Z. Alghareeb, S. Walton, and J. Williams. Well Placement Optimization Under Constraints Using Modified Cuckoo Search. Paper SPE-SAS-353 presented at the SPE-SAS Annual Technical Symposium & Exhibition, Al Khobar, Saudi Arabia, 2014.

[10] A. Alhuthali, D. Oyerinde, and A. Datta-Gupta. Optimal Waterflood Management Using Rate Control. *SPE Reservoir Evaluation & Engineering*, **10**(5): 539–551, 2007.

[11] A. Alhuthali, A. Datta-Gupta, B. Yuen, and J. Fontanilla. Field Application of Waterflood Optimization via Optimal Rate Control with Smart Wells. Paper SPE 118948 presented at the SPE Reservoir Simulation Symposium, Woodlands, Texas, USA, 2009.

[12] L. Almeida, Y. Túpac, G. Lazo, M. Pacheco, and M. Marley. Evolutionary Optimization of Smart-Wells Control Under Technical Uncertainties. Paper SPE 107872 presented at the SPE Latin American and Caribbean Petroleum Engineering Conference, Buenos Aires, Argentina, 2007.

[13] A. Antoulas and D. Sorensen. Approximation of Large-scale Dynamical Systems: An Overview. *International Journal of Applied Mathematics and Computer Science*, **11**(5): 1093–1121, 2001.

[14] E. Arian, M. Fahl, and E. Sachs. Trust-Region Proper Orthogonal Decomposition for Flow Control. 2000.

[15] P. Astrid, S. Weiland, K. Willcox, and T. Backx. Missing Point Estimation in Models Described by Proper Orthogonal Decomposition. *IEEE Transactions on Automatic Control*, **53**(10): 2237–2251, 2008.

[16] C. Audet and J. Dennis. A Pattern Search Filter Method for Nonlinear Programming without Derivatives. *SIAM Journal on Optimization*, **14**(4): 980–1010, 2004.

[17] C. Audet and J. Dennis. Mesh Adaptive Direct Search Algorithms for Constrained Optimization. *SIAM Journal on Optimization*, **17**(1): 188–217, 2006.

[18] K. Aziz and A. Settari. *Petroleum Reservoir Simulation.* Applied Science Publishers, 1979.

[19] W. Bangerth, H. Klie, M. Wheeler, P. Stoffa, and M. Sen. On Optimization Algorithms for The Reservoir Oil Well Placement Problem. *Computational Geosciences*, **10**(3): 303–319, 2006.

[20] M. Barrault, Y. Maday, N. Nguyen, and A. Patera. An Empirical Interpolation Method: Application to Efficient Reduced-Basis Discretization of Partial Differential Equations. *Comptes Rendus Mathematique*, **339**(9): 667–672, 2004.

[21] A. Bittencourt and R. Horne. Reservoir Development and Design Optimization. Paper SPE 38895 presented at the SPE Annual Technical Conference and Exhibition, San Antonio, Texas, USA, 1997.

[22] B. Bond. *Stability-Preserving Model Reduction for Linear and Nonlinear Systems Arising in Analog Circuit Applications by.* PhD thesis, Massachusetts Institute of Technology, 2010.

[23] D. Bratton and J. Kennedy. Defining a Standard for Particle Swarm Optimization. In *2007 IEEE Swarm Intelligence Symposium*, 120–127, 2007.

[24] D. Brouwer and J. Jansen. Dynamic Optimization of Waterflooding With Smart Wells Using Optimal Control Theory. *SPE Journal*, **9**(4): 29–31, December 2004.

[25] M. Buffoni and K. Willcox. Projection-based Model Reduction for Reacting Flows. In *Proceedings of AIAA 40th Fluid Dynamics Conference and Exhibit*, Chicago, Illinois, USA, 2010.

[26] T. Bui-Thanh and K. Willcox. Model Reduction for Large-Scale CFD Applications Using the Balanced Proper Orthogonal Decomposition. In *17th AIAA Computational Fluid Dynamics Conference*, Toronto, Canada, 2005.

[27] T. Bui-Thanh, M. Damodaran, and K. Willcox. Aerodynamic Data Reconstruction and Inverse Design Using Proper Orthogonal Decomposition. *AIAA Journal*, **42**(8): 1505–1516, 2004.

[28] M. Cardoso. *Development and Application of Reduced-Order Modeling Procedures for Reservoir Simulation.* PhD thesis, Department of Energy Resources Engineering, Stanford University, 2009.

[29] M. Cardoso and L. Durlofsky. Use of Reduced-Order Modeling Procedures for Production Optimization. Paper SPE 119057 PP presented at the SPE Reservoir Simulation Symposium, Woodlands, Texas, USA, 2009.

[30] M. Cardoso and L. Durlofsky. Linearized Reduced-Order Models for Subsurface Flow Simulation. *Journal of Computational Physics*, **229**(3): 681–700, 2010.

[31] S. Chaturantabut. *Nonlinear Model Reduction via Discrete Empirical Interpolation.* PhD thesis, Rice University, 2011.

[32] S. Chaturantabut and D. Sorensen. Nonlinear Model Reduction via Discrete Empirical Interpolation. *SIAM Journal on Scientific Computing*, **32**(5): 2737–2764, January 2010.

[33] S. Chaturantabut and D. Sorensen. Application of POD and DEIM on Dimension Reduction of Nonlinear Miscible Viscous Fingering in Porous Media. *Mathematical and Computer Modeling of Dynamical Systems*, **17**(4): 337–353, 2011.

[34] M. Christie and M. Blunt. Tenth SPE Comparative Solution Project : A Comparison of Upscaling Techniques. *SPE Reservoir Evaluation and Engineering*, **4**: 308–317, 2001.

[35] A. Conn, N. Gould, and Toint. P. Trust-Region Methods. *MPS-SIAM Series on Optimization*, 2000.

[36] D. Bertsekas. *Constrained Optimization and Lagrange Multipliers Mehotds.* Athena Scientific, Belmont, Massachusetts, 1982.

[37] J. Dennis and R. Schnabel. *Numerical Methods for Unconstrained Optimization an Nonlinear Equations.* Princton Hall, Engelwood Cliffs, NJ, 1983.

[38] I. Dones, S. Skogestad, and H. Preisig. Application of Balanced Truncation to Nonlinear Systems. *Industrial & Engineering Chemistry Research*, **50**(17): 10093–10101, 2011.

[39] M. Farshi. *Improving Genetic Algorithm For Optimum Well Placement.* Master's thesis, Department of Energy Resourses Engineering, Stanford University, 2008.

[40] R. Fletcher and S. Leyffer. Nonlinear Programming without A Penalty Function. *Mathematical Programming*, **91**(2): 239–269, January 2002.

[41] R. Fletcher, S. Leyffer, and P. Toint. A Brief History of Filter Methods. Technical report, Mathematics and Computer Science Division, Argonne National Laboratory, 2006.

[42] Institute for Energy Research. Fossil fuels still king in eia's annual energy outlook 2013, 2012. URL `http://instituteforenergyresearch.org/analysis/fossil-fuels-still-king-in-eias-annual-energy-outlook-2013/`.

[43] F. Forouzanfar, G. Li, and A. Reynolds. A Two-Stage Well Placement Optimization Method Based on Adjoint Gradient. Paper SPE 135304 presented at the SPE Annual Technical Conference and Exhibition, Florence, Italy, 2010.

[44] D. Galbally, K. Fidkowski, K. Willcox, and O. Ghattas. Non-linear Model Reduction for Uncertainty Quantification in Large-Scale Inverse Problems. *International Journal for Numerical Methods in Engineering*, **81**(12): 1581–1608, 2010.

[45] A. Gandomi, S. Talatahari, X. Yang, and S. Deb. Design Optimization of Truss Structures Using Cuckoo Search Algorithm. *The Structural Design of Tall and Special Buildings*, **22**(17): 1330–1349, 2013.

[46] D. Gratton. *Reduced-Order, Trajectory Piecewise-Linear Models for Nonlinear Computational Fluid Dynamics.* Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, 2004.

[47] M. Grepl, Y. Maday, N. Nguyen, and A. Patera. Efficient Reduced-basis Treatement of Nonaffine and Nonlinear Partial Differential Equations. *Mathematical Modeling and Numerical Analysis*, **41**(3): 575–605, 2007.

[48] S. Gugercin and A. Antoulas. A Survey of Model Reduction by Balanced Truncation and Some New Results. *International Journal of Control*, **77**(8): 748–766, 2004.

[49] B. Guyaguler. *Optimization of Well Placement and Assessment of Uncertainty.* PhD thesis, Department of Petroleum Engineering, Stanford University, 2002.

[50] B. Guyaguler and R. Horne. Uncertainty Assessment of Well-Placement Optimization. *SPE Journal*, **7**(1): 24–32, 2004.

[51] R. Hassan, B. Cohanim, O. deWeck, and G. Venter. A comparison of Particle Swarm Optimization and The Genetic Algorithm. *American Institute of Aeronautics and Astronautics*, 1–13, 2004.

[52] J. He. *Enhanced Linearized Reduced-Order Models for Subsurface Flow Simulation.* Master's thesis, Department of Energy Resources Engineering, Stanford University, 2010.

[53] J. He and L. Durlofsky. Reduced-Order Modeling for Compositional Simulation Using Trajectory Piecewise Linearization. Paper SPE 163 presented at the SPE Reservoir Simulation Symposium, The Woodlands, Texas, USA, 2013.

[54] J. He, J. Saetrom, and L. Durlofsky. Enhanced Linearized Reduced-Order Models for Subsurface Flow Simulation. *Journal of Computational Physics*, **230**: 8313–8341, 2011.

[55] T. Heijn, R. Markovinovic, and J. Jansen. Generation of Low-Order Reservoir Models Using System-Theoretical Concepts. *SPE Journal*, **2**(9): 202–218, 2004.

[56] A. Hochman, B. Bond, and J. White. A Stabilized Discrete Empirical Interpolation Method for Model Reduction of Electrical, Thermal, and Microelectromechanical Systems. In *Proceedings of the 48th Design Automation Conference*, San Diego, California, USA, 2011. ACM Press.

[57] V. Hung and H. Tran. Modeling and Control of Physical Processes Using Proper Orthogonal Decomposition. *Mathematical and Computer Modeling*, **33**: 223–236, 2001.

[58] O. Isebor. *Constrained Production Optimization with an Emphasis on Derivative-Free Methods.* Master's thesis, Department of Energy Resources Engineering, Stanford University, 2009.

[59] O. Isebor. *Derivative-Free Optimizaiton for Generalized Oil Field Development.* PhD thesis, Department of Energy Resources Engineering, Stanford University, 2013.

[60] O. Isebor. Derivative-Free Generalized Field Development Optimization. Paper SPE 167633-STU presented at the SPE Reservoir Simulation Symposium, New Orleans, Louisiana, USA, 2013.

[61] O. Isebor, L. Durlofsky, and D. Echeverría. A Derivative-Free Methodology with Local and Global Search for The Constrained Joint Optimization of Well Locations and Controls. *Computational Geosciences*, 1–20, 2013.

[62] O. Isebor, D. Echeverría, and L. Durlofsky. Generalized Field Development Optimization Using Derivative-Free Procedures. Paper SPE 163631 MS presented at the SPE Reservoir Simulation Symposium, The Woodlands, Texas, USA, 2013.

[63] O. Izgec, M. Sayarpour, and G. Shook. Optimizing Volumetric Sweep Efficiency in Waterfloods by Integrating Streamlines, Design of Experiments, and Hydrocarbon F-Φ Curves. Paper SPE 132609 presented at the Western North America Regional Meeting, Anaheim, CA, USA, 2010.

[64] I. Kalashnikova and M. Barone. Efficient Non-Linear Proper Orthogonal Decomposition (POD)/ Galerkin Reduced Order Models with Stable Penalty Enforcement of Boundary Conditions. *International Journal for Numerical Methods in Engineering*, **00**: 1–28, 2011.

[65] A. Kellems, S. Chaturantabut, D. Sorensen, and S. Cox. Morphologically Accurate Reduced-Order Modeling of Spiking Neurons. *Journal of computational neuroscience*, **28**(3): 477–94, 2010.

[66] J. Kraaijevanger, P. Egberts, J. Valstar, and H. Buurman. Optimal Waterflood Design Using the Adjoint Method. Paper SPE 105764 presented at the SPE Reservoir Simulation Symposium, Houston, Texas, USA, 2007.

[67] S. Le Digabel. Algorithm 909: NOMAD: Nonlinear Optimization with the MADS Algorithm. *ACM Transactions on Mathematical Software*, **37**(4): 1–15, February 2011.

[68] R. Li, A. Reynolds, and D. Oliver. Production Data. *SPE Journal*, **8**(4): 328–340, 2003.

[69] K. Lie, S. Krogstad, I. Ligaarden, J. Natvig, H. Nilsen, and B. Skaflestad. Open Source MATLAB Implementation of Consistent Discretisations on Complex Grids. *Computational Geosciences*, **16**(2): 297–322, 2011.

[70] K. Lim, B. Khoo, and K. Willcox. Model Order Reduction for Determining Bubble Parameters to Attain a Desired Fluid Surface Shape. **36**: 2–5, 2007.

[71] A. Lu and E. Wachspress. Solution of Lyapunov Equation by Alternating Direction Implicit Iteration. *Computers & Mathematics with Applications*, **21**(9): 43–58, 1991.

[72] R. Markovinović and J. Jansen. Accelerating Iterative Solution Methods Using Reduced-Order Models as Solution Predictors. *International Journal for Numerical Methods in Engineering*, **68**(5): 525–541, 2006.

[73] MATLAB Release 2013b. *The MathWorks Inc.* Natick, Massachusetts, USA, 2013.

[74] B. Moore. Principal Component Analysis in Linear Systems: Controllability, Observability, and Model Reduction. *IEEE Transactions on Automatic Control*, **26**(1): 17–32, 1981.

[75] N. Nguyen and J. Peraire. An Efficient Reduced-Order Modeling Approach for Non-linear Parametrized Partial Differential Equations. *International Journal for Numerical Methods in Engineering*, **73**: 27–55, 2008.

[76] N. Nguyen, A. Patera, and J. Peraire. A Best Points Interpolation Method for Efficient Approximation of Parametrized Functions. *International Journal for Numerical Methods in Engineering*, **73**: 521–543, 2008.

[77] J. Nocedal and S. Wright. *Numerical Optimization.* Springer, second edi edition, 2006.

[78] Y. Notay. An Aggregation-based Algebraic Multigrid Method. *Electronic Transactions on Numerical Analysis*, **37**: 123–146, 2010.

[79] J. Onwunalu. *Optimization of Field Development Using Particle Swarm Optimization and New Well Pattern Optimization.* PhD thesis, Department of Energy Resources Engineering, Stanford University, 2010.

[80] U. Ozdogan and R. Horne. Optimization of Well Placement Under Time-Dependent Uncertainty. *SPE Reservoir Evaluation & Engineering*, **9**(2): 135–145, 2006.

[81] Y. Pan and R. Horne. Improved Methods for Multivariate Optimization of Field Development Scheduling and Well Placement Design. Paper SPE 49055 presented at the SPE Annual Technical Conference and Exhibition, New Orleans, Lousiana, USA, 1998.

[82] D. Peaceman. Interpretation of Well-Block Pressures in Numerical Reservoir Simulation. *SPE Journal*, **23**(3): 531–543, 1983.

[83] B. Peherstorfer, D. Butnaru, K. Willcox, and H. Bungartz. Localized Discrete Empirical Interpolation Method. Technical Report June, MIT Aerospace Computational Design Laboratory, 2013.

[84] M. Rewienski. *A Trajectory Piecewise-Linear Approach to Model Order Reduction of Nonlinear Dynamical Systems.* PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 2003.

[85] C. Rowley. Model Reduction for Fluids Using Balanced Proper Orthogonal Decomposition. *International journal of Bifurcation and Chaos*, **3**(15): 997–1013, 2005.

[86] J. Saetrom. *Reduction of Dimensionality in Spatiotemporal Models.* PhD thesis, Department of Mathematical Sciences, Norwegian University of Science and Technology, 2010.

[87] C. Sainvitu. *Filter-Trust-Region Methods for Nonlinear Optimization.* PhD thesis, 2007.

[88] P. Sarma. *Efficient Closed-Loop Optimal Control of Petroleum Reservoirs Under Uncertainty.* PhD thesis, Department of Petroleum Engineering, Stanford University, 2006.

[89] P. Sarma and W. Chen. Efficient Well Placement Optimization With Gradient-Based Algorithms and Adjoint Models. Paper SPE 112257 presented at the SPE Intelligent Energy Conference and Exhibition, Amsterdam, The Netherlands, 2008.

[90] G. Selvi and T. Purusothaman. Cryptanalysis of Simple Block Ciphers Using Extensive Heuristic Attacks. *European Journal of Scientific Research*, **78**: 198–221, 2012.

[91] R. Smith. Matrix Equation XA + BX = C*. *SIAM Journal of Applied Mathimatics*, **16**(1): 198–201, 1968.

[92] J. Spall. An Overview of the Simultaneous Perturbation Method for Efficient Optimization. *John Hopkins Applied Physics Laboratory Technical Digest*, **19** (4): 482–492, 1998.

[93] E. Suwartadi. *Gradient-based Methods for Production Optimization of Oil Reservoirs.* PhD thesis, Mathematics and Electrical Engineering, Norwegian University of Science and Technology, 2012.

[94] A. Tan Yong Kwang. *Reduced Basis Method for 2nd Order Wave Equation : Application to One-Dimensional Seismic Problem by.* Master's thesis, School of Engineering, Massachusetts Institute of Technology, 2005.

[95] J. van Doren, R. Markovinović, and J. Jansen. Reduced-Order Optimal Control of Water Flooding Using Proper Orthogonal Decomposition. *Computational Geosciences*, **10**(1): 137–158, 2006.

[96] D. Vasilyev, M. Rewienski, and J. White. A TBR-based Trajectory Piecewise-Linear Algorithm for Generating Accurate Low-order Models for Nonlinear Analog Circuits and MEMS. In *Proceedings of the 40th Conference on Design Automation,*, 490–495, 2003.

[97] A. Vaz and L. Vicente. A Particle Swarm Pattern Search Method for Bound Constrained Global Optimization. *Journal of Global Optimization*, **39**(2): 197–219, 2006.

[98] R. Vazquez. Training Spiking Neural Models Using Cuckoo Search Algorithm. In *2011 IEEE Congress of Evolutionary Computation*, 2011.

[99] A. Vendl, H. Faß bender, G. Stefan, and M. Mifsud. Model Order Reduction for Steady Aerodynamics of High-Lift Configurations. Technical report, 2013.

[100] A. Wachter and L. Biegler. Line Search Filter Methods for Nonlinear Programming: Local Convergence. *SIAM Journal on Optimization*, **16**(1): 32–48, August 2005.

[101] A. Wachter and L. Biegler. Line Search Filter Methods for Nonlinear Programming: Motivation and Global Convergence. *SIAM Journal on Optimization*, **16** (1): 1–31, 2005.

[102] S. Walton. *Gradient Free Optimisation in Selected Engineering Applications.* PhD thesis, College of Engineering, Swansea University, 2013.

[103] S. Walton, O. Hassan, K. Morgan, and M. Brown. Modified Cuckoo Search: A New Gradient Free Optimisation Algorithm. *Chaos, Solitons & Fractals*, **44** (9): 710–718, 2011.

[104] H. Wang, D. Echeverría, L. Durlofsky, and A. Cominelli. Optimal Well Placement Under Uncertainty Using a Retrospective Optimization Framework. *SPE Journal*, **17**(1): 112–121, 2012.

[105] K. Willcox. Multidisciplinary design and optimization. 2012.

[106] X. Yang and S. Deb. Cuckoo Search via Le'vy Flights. In *Proceedings of World Congress on Nature and biologically Inspired Computing*, 210–214, India, 2009. IEEE Publications.

[107] B. Yeten. *Optimum Deployment of Nonconventional Wells.* PhD thesis, Department of Petroleum Engineering, Stanford University, 2003.

[108] B. Yeten, L. Durlofsky, and K. Aziz. Optimization of Smart Well Control. Paper SPE 79031 presented at the SPE International Thermal Operations and Heavy Oil Symposium and International Horizontal Well Technology Conference, Calgary, Alberta, Canada, 2002.

[109] A. Yildiz. Cuckoo Search Algorithm for The Selection of Optimal Machining Parameters in Milling Operations. *The International Journal of Advanced Manufacturing Technology*, **64**(1-4): 55–61, 2012.

[110] S. Yoon, Z. Alghareeb, and J. Williams. Development of Reduced-order Oil Reservoir Models using Localized DEIM. Paper SPE 170741-MS presented at the SPE Annual Technical Conference & Exhibition, Amsterdam, The Netherlands, 2014.

[111] M. Zandvliet. *Model-based Lifecycle Optimization of Well Locations and Production Settings in Petroleum Reservoirs.* PhD thesis, Delft University of Technology, 2008.

[112] M. Zandvliet, M. Handels, G. van Essen, D. Brouwer, and J. Jansen. Adjoint-Based Well-Placement Optimization Under Production Constraints. *SPE Journal*, **13**(4): 26–28, 2008.

[113] D. Zheng, K. Hoo, and M. Piovoso. Low-order Model Identification of Distributed Parameter Systems by a Combination of Singular Value Decomposition and The Karhunen-Loeve Expansion. *Industrial & Engineering Chemistry Research*, **41**: 1545–1556, 2002.

Nomenclature

Abbreviations

ared	actual function reduction
B&B	branch and bound
BHP	bottom-hole pressure
BPIM	best point interpolation method
CS	cuckoo search
DE	differential evolution
DEIM	discrete empirical interpolation method
E&P	exploration and producing
EIM	empirical interpolation method
FD	finite difference
FOM	full-order model
GA	genetic algorithms
GPS	general pattern search
ICV	inflow control valve
IPOPT	interior point optimizer
LHS	latin hyper-cube sampling
LTI	linear time invariant
MADS	mesh adaptive direct search

MCS	modified cuckoo search
MCS-MADS	hybrid of modified cuckoo search and mesh adaptive direct search
MPE	missing point estimation
MRST	MATLAB reservoir simulation toolbox
NLP	nonlinear programming
NPV	net present value
POD	proper orthogonal decomposition
pred	predicted function reduction
PSO	particle swarm optimization
PVI	pore volume injected
ROM	Reduced-order model
SA	simulated annealing
SLI	simplex linear interpolation
SPSA	simultaneous perturbation stochastic approximation
SQP	sequential quadratic programming
STB	stock tank barrel
SVD	singular value decomposition
TBR	truncated-balance reduction
TOF	time-of-flight
TPWL	trajectory piecewise linearization
WPO	well pattern optimization

Variables

$\mathbf{B}$	controls arrangement matrix
$\mathbf{c}$	vector of nonlinear constraint functions
$\mathbf{F}$	total flux vector
$\mathbf{G}$	gravitational forces vector
$\mathbf{g}$	residual vector

$\mathbf{G}_c$	controllability Grammian
$\mathbf{G}_o$	observability Grammian
$\mathbf{I}$	identity matrix
$\mathbf{J}$	Jacobian matrix
$\mathbf{K}$	absolute permeability tensor
$\mathbf{p}$	pressure vector
$\mathbf{Q}$	source/sink vector
$\mathbf{T}$	transmissibility matrix
$\mathbf{u}$	input control vector
$\mathbf{v}$	Darcy velocity
$\mathbf{W}_L$	left transformation matrix (from TBR)
$\mathbf{W}_R$	right transformation matrix (from TBR)
$\mathbf{x}$	poll center iterate
$\mathcal{F}$	filter set
$\mathbb{F}$	flux snapshots
$\mathbb{P}$	pressure snapshot matrix
$\mathbb{S}$	saturation snapshots
$\mathbf{L}_\mathrm{B}$	optimization variable lower bound
$\mathbf{U}_\mathrm{B}$	optimization variable upper bound
$\tilde{\mathbf{B}}$	reduced arrangement matrix
$\tilde{\mathbf{G}}$	reduced gravitational force vector
$\tilde{\mathbf{P}}$	reduced pressure vector
$\tilde{\mathbf{T}}$	reduced transmissibility matrix
D	poll spanning direction matrix
f_w	water fractional flow
G	generation number
g	gravitational acceleration constant

J	optimization objective function
k_r	relative permeability
M	set of poll trial points
n_D	poll spanning directions
p	pressure
p_a	egg discovery probability
q	flow rate
r_w	wellbore radius
S	saturation
t	time
z	depth
WI	well index

Greek Symbols

α	schedule interpolation parameter or Lèvy flight step size
χ	merit function
Δ^m	MADS mesh size
Δ^p	MADS poll size
ϵ	tolerance
η	poll size contraction parameter
κ	calibration parameter for well placement
λ	phase mobility
$\mathbf{\Phi}_L$	pressure left basis matrix
$\mathbf{\Phi}_R$	pressure right basis matrix
$\mathbf{\Psi}$	flux basis matrix
$\mathbf{\Upsilon}$	saturation basis matrix
μ	viscosity
Ω	control volume or bounded set of all allowed optimization variables

ϕ	porosity
$\boldsymbol{\delta}$	trial step increment for well control
ψ	mesh size contraction parameter
ρ	density or calibration parameter for well controls
σ	singular value or standar deviation
Θ	relative omitted energy
$\triangle$	trust-region size for well control
φ	golden ratio
$\wp$	penalty parameter
h	aggregated constraint violation function
L_c	Lorenz coefficient

Subscripts

0	initial value
$\vec{\mathbf{z}}$	interpolation indices vector
i	spatial index
j	spatial index or phase type
k	iteration index
o	oil phase
r	reduced representation
t	total
w	water phase

Superscripts

$\mathcal{R}$	refer to function evaluated using ROM
F	feasibility indicator
I	infeasibility indicator
n	time step, full-order dimension
w	well