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

THE GRADUATE SCHOOL

APPLICATION OF SPSA-TYPE ALGORITHMS TO PRODUCTION OPTIMIZATION

by
Sy Thanh Do

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

The Graduate School
The University of Tulsa

2012

THE UNIVERSITY OF TULSA

THE GRADUATE SCHOOL

APPLICATION OF SPSA-TYPE ALGORITHMS TO
PRODUCTION OPTIMIZATION

by
Sy Thanh Do

A DISSERTATION
APPROVED FOR THE DISCIPLINE OF
PETROLEUM ENGINEERING

ABSTRACT

Sy Thanh Do (Doctor of Philosophy in Petroleum Engineering)

Application of SPSA-Type Algorithms to Production Optimization

Directed by Albert Reynolds and Gaoming Li

156 pp., Chapter 5: Conclusions

(565 words)

Our objective is to devise an efficient optimization algorithm for estimating optimal well controls during the production optimization step of closed-loop reservoir management where the optimization algorithm does not require an adjoint method for the calculation of derivatives. Research within the TUPREP group strongly suggests that gradient-based algorithms will be the most efficient and reliable methods for estimating well controls to maximize the net present value (NPV), or cumulative oil recovery of an oil field. However, most major oil companies use commercial reservoir simulators to make future predictions of production, and such commercial simulators generally lack efficient gradient-based production optimization algorithms where computational efficiency demands that the gradient of the cost function (e.g., NPV or cumulative oil production over the life of the field) be computed by an adjoint method. Adding an adjoint solution without access to the source code is not feasible. Therefore, it is important to implement optimization algorithms that can be easily coupled with a commercial simulator simply by using the simulator as a black box. While many such gradient-free optimization algorithms are available, based on the excellent performance of gradient-based methods, it is desirable to develop a procedure that mimics the property of methods that use a gradient. One such method is the simultaneous perturbation stochastic approximation (SPSA) method which estimates a stochastic gradient by a procedure that is far more efficient than estimating the derivative with a finite difference approximation.

The purpose of this study is to develop a computationally efficient version of the SPSA algorithm for the constrained production optimization problem involved in the optimal control step of closed-loop reservoir management. The SPSA is a stochastic optimization algorithm, which obtains its stochastic gradient by simultaneously perturbing all components of the well control vector in a stochastic way. Although the SPSA gradient is stochastic, it is always uphill for a sufficiently small perturbation size and its expectation is equal to the true gradient as the perturbation size goes to zero. The most challenging problem in production optimization is to honor state-control constraints which are nonlinear. The SPSA algorithm has been applied to production optimization problems with only bound constraints and implementations used in previous published work were computationally inefficient. In this study, the focus will be developing and implementing a computationally efficient SPSA algorithm for production optimization problems with general equality, inequality and bound constraints. Here, the bound constraints are converted into the new unbounded control variables using a log-transformation. Although the results are not shown in this work, simply truncating control variables at their bounds consistently yields slightly lower NPVs values than are obtained with the log-transform method used here. All constraints are incorporated into the objective function to be maximized using the augmented Lagrangian function. The advantage of this method is that the general inequality and equality constraints can conceptually be handled using suitable values of the penalty parameter and Lagrange multipliers. In addition to the basic SPSA algorithm, which uses samples from the symmetric Bernoulli distribution as perturbations, a SPSA type algorithm that uses samples from Gaussian distribution is also tested. To improve the efficiency of the SPSA algorithm, a combined search direction SPSA algorithm is proposed, in which the search direction is a combination of the current SPSA gradient and the “best” SPSA gradient or some “good” former SPSA gradients calculated in the last few iterations.

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Albert C. Reynolds, Jr., Professor of Petroleum Engineering and Mathematical Science of The University of Tulsa, and Dr. Gaoming Li, Associate Professor of Petroleum Engineering of The University of Tulsa, for their support, guidance, insights and encouragement during my five years' Ph.D. study at The University of Tulsa. I specially thank Dr. Mauricio Prado, Associate Professor of Petroleum Engineering and Mathematical Science of The University of Tulsa, and Dr. William Coberly, Associate Professor of Mathematical Science of The University of Tulsa , for serving on my committee and for their helpful suggestions and comments. I greatly appreciate the help of Mrs. Judy Teal and Mrs. Loreta M. Watkins during my study in The University of Tulsa. I would like to extend my thanks and appreciations to all the other faculty members of the Petroleum Engineering department of The University of Tulsa for their guidance through my courses of study as a graduate student.

I gratefully acknowledge financial support from the member companies of the University of Tulsa Petroleum Reservoir Exploitation Projects (TUPREP). I would like to express my gratitude to the Vietnam International Education Development Scholarship Program (VIED) for the generous financial support for the research.

I am thankful to my wife and my babies for their unconditional love and support over the last few years.

This work is dedicated to my parents, my wife and my babies for their unconditional love and support.

TABLE OF CONTENTS

LIST OF TABLES

Page

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background and Literature Review

Production optimization is a process to identify the optimal well operating conditions in order to maximize total hydrocarbon recovery over the life of a reservoir or to maximize net present value (NPV) of production. Production optimization is an essential element in closed-loop reservoir management, which alternates production optimization with data assimilation to form a real-time reservoir management strategy (Brouwer and Jansen, 2004; Jansen et al., 2005; Sarma et al., 2005, 2008; Chen et al., 2009; Wang et al., 2009; Peters et al., 2009; Chen et al., 2010). Closed-loop or real-time reservoir management alternates data assimilation with a step in which optimal well controls are estimated. Starting with an initial reservoir model (at time zero), one estimates how to operate production and injection wells on each of N_C predefined control steps to maximize NPV or some other cost function. During the first control step, one operates the wells at the estimated optimal controls (well pressures and/or rates) and during production for this first control step, one collects production data and updates the reservoir model(s) by assimilating (history-matching) these data. At the end of the first control step, based on the updated model(s), one estimates the optimal well controls for the remaining $N_C - 1$ control steps by again maximizing NPV (or some other cost function) using an optimization algorithm. Then we produce the field during the time interval corresponding to the second control step, and during this production, we again update the reservoir model(s) by assimilating measured data. This process of alternating data assimilation with production optimization by estimating optimal well controls is continued throughout the life of the reservoir.

Our work focuses on the production optimization step of closed-loop reservoir management. In the production optimization literature, well controls are often adjusted using the gradient as the search direction. The gradient can be the true gradient obtained through the adjoint procedure (Brouwer and Jansen, 2004; Jansen et al., 2005; Sarma et al., 2005, 2008; Chen et al., 2010), a stochastic gradient (Wang et al., 2009), or an ensemble-derived average gradient (Chen et al., 2009). It can be shown that the ensemble-derived gradient is very similar to the well-known simplex gradient (Zhao et al., 2011). To obtain the simplex gradient, one first predicts a set of NPV values using a set of different control vectors. If the number of control vectors is the same as the dimension of the control vector, the simplex gradient can be easily calculated by solving a linear system of equations, as long as the set of control vectors is linearly independent. If the relationship between the NPV and the control vector is truly linear, the calculated simplex gradient will be the true gradient (Zhao et al., 2011). However, the number of control vectors is usually far less than the dimension of the control vector (or gradient), and in this case the simplex gradient calculation is performed with a singular value decomposition procedure (Bortz and Kelley, 1998; Kelley, 1999; Tseng, 1999; Conn et al., 2006; Custsódio and Vicente, 2007).

In reality, the NPV has a nonlinear relationship with the control vector, so the simplex gradient or ensemble-derived gradient can only give a rough estimate of the true gradient. The efficiency of a simplex gradient-based optimization algorithm depends on the number of control vectors used for the simplex gradient calculation. In general, algorithms based on the adjoint gradient provide the most computationally efficient optimization algorithms, as the gradient of NPV with respect to all the control variables can be computed in one backward adjoint run, which takes about one third of the time required by one forward simulation run. Unfortunately, the development of an adjoint code requires detailed knowledge of the reservoir simulator numerics, and an adjoint gradient calculation is only available in limited commercial reservoir simulation software for a small set of reservoir processes and in some in-house simulators at some major oil companies and research institutes.

Another way of calculating the true gradient is to use the finite-difference method, which perturbs one control variable at a time. This method requires a forward simulation run for each perturbation, so the method is very time consuming if the number of control variables is large. An alternative to the finite-difference calculation of the gradient is the simultaneous perturbation method used in Simultaneous Perturbation Stochastic Approximation (SPSA) (Spall, 1992, 1998, 2003; Wang and Spall, 2008). Instead of perturbing one control variable at a time as in the finite-difference method, the SPSA algorithm perturbs the whole control vector using a random vector of perturbations. The SPSA gradient is hence a stochastic gradient instead of the true gradient. However, the expectation of the SPSA gradient is the true gradient as the perturbation size goes to zero, and the search direction of the SPSA gradient is always an uphill direction for a sufficiently small perturbation. Li and Reynolds (2011) proposed calculating an SPSA gradient using a Gaussian perturbation vector sampled from a normal distribution with mean equal to the dimensional zero vector and covariance matrix. The modified SPSA algorithm based on Gaussian perturbations have been successfully applied in a variety of petroleum engineering applications (Li and Reynolds, 2011; Zhao et al., 2011).

In our work, the use of the augmented Lagrangian function as the objective function is proposed to maximize a constrained production optimization problem with the SPSA algorithm. To deal with the bound constraints, the log-transformation is applied to convert the optimization problem with bound constraints into one without bound constraints. The augmented Lagrangian function which incorporates all the equality and inequality constraints, is suitable for large-scale optimal well control problems, and can easily handle nonlinear control-state constraints. Moreover, the method is easy to implement and easy to couple with any commercial reservoir simulator as long as gradients can be approximated outside of the reservoir simulator. In the results presented here, we consider only the two phase flow of oil and water.

1.2 Problem formulation

To solve production optimization problems, we maximize NPV by adjusting well controls. For a two-phase (oil and water) flow reservoir under water-flooding, NPV is defined as:

$$J(u,y) = \sum_{n=1}^{N} \left[\sum_{i=1}^{N_{\text{prod}}} (r_{\text{o}} q_{\text{o},i}^{n} - r_{\text{w}} q_{\text{w},i}^{n}) - \sum_{i=1}^{N_{\text{winj}}} r_{\text{winj}} q_{\text{winj},i}^{n} \right] \frac{\Delta t^{n}}{(1+b)^{t^{n}/365}}, \tag{1.1}$$

where u is the n_u-dimensional column vector containing all well controls at all time steps; $y = y(u, m)$ denotes the vector of primary variables solved for during the simulation run; y is a function of both the vector of all well controls u, as well as a function of the reservoir model m, which is assumed known or at least fixed during production optimization; N is the total number of reservoir simulation time steps; N_{prod} is the total number of producers; N_{winj} is the total number of water injection wells; r_{o} is the oil revenue (\$/STB); r_{w} is the water production cost (\$/STB); r_{winj} is the water injection cost (\$/STB); $q_{\text{o},i}^{n}$ is the average oil production rate of the i^{th} producer (STB/D) during the n^{th} time step; $q_{\text{w},i}^{n}$ is the average water production rate of the i^{th} producer (STB/D) during the n^{th} time step; $q_{\text{winj},i}^{n}$ is the average water injection rate of the i^{th} injection well (STB/D) during the n^{th} time step; b is the annual discount rate; t^n is the cumulative time (days) up to the n^{th} time step; and Δt^n (days) is the time interval of the n^{th} simulation time step.

Because well rates depend on the vector of control variables, u, the NPV defined in Eq. 1.1 is a function of the well control vector u and the dynamic state vector y, which is the vector of variables solved by the reservoir simulator, e.g., pressures and saturations. The control variables may include water injection rates or the producer bottomhole pressure (BHP) or/and other type of rate. The maximization of the NPV in Eq. 1.1 is usually subject to some equality, inequality, and bound constraints given, respectively, as:

$$e_j(u, y) = 0,\ j = 1, .., n_{\mathrm{e}}, \tag{1.2}$$

$$c_j(u, y) \leq 0,\ j = 1, .., n_{\mathrm{ine}}, \tag{1.3}$$

and

$$u_i^{\mathrm{low}} \leq u_i \leq u_i^{\mathrm{up}},\ i = 1, 2, ..., n_{\mathrm{u}}. \tag{1.4}$$

In production optimization, common equality constraints (Eq. 1.2) are the field rate constraints, where the field water injection rate and the field total liquid production rate are fixed at constant values. Common inequality constraints (Eq. 1.3) might be due to the physical limitations of the production and injection facilities. For example, the water treatment facility might be able to provide only a certain amount of injection each day at maximum, or the capacity of the separation facilities might limit the total liquid production from the field. When a well is operated on rate control, the lower bound for the rate is zero, which corresponds to shut-in. When the well is producing under BHP control, we may wish to keep the BHP above the bubble point pressure or above the minimum BHP required to produce fluid to the surface. For injection wells, it is normally preferable to have the injection BHP lower than the parting pressure to control sweep efficiency.

1.2.1 Simple bound constraints

One way to deal with the upper and lower bounds of the control variables is to use a log-transformation. This method guarantees that the control variables always remain within the upper and lower bounds during optimization (Gao et al., 2007; Wang et al., 2009; Zhao et al., 2011). In the log-transformation method, the new variable s_i is defined as:

$$s_i = \ln\left(\frac{u_i - u_i^{\text{low}}}{u_i^{\text{up}} - u_i}\right),\ i = 1, 2, ..., n_{\text{u}}. \tag{1.5}$$

As the control variable u_i approaches its lower bound u_i^{low}, the transformed variable s_i approaches $-\infty$, and as u_i approaches its upper bound u_i^{up}, s_i approaches $+\infty$. By using this log-transformation, a simple bound constrained problem can be transformed into an unconstrained optimization problem. When the log-transformation is applied during optimization, all operations are done in the transformed domain, and the actual control variables are obtained using the inverse log-transformation:

$$u_i = \frac{\exp(s_i)u_i^{\text{up}} + u_i^{\text{low}}}{1 + \exp(s_i)} = \frac{u_i^{\text{up}} + u_i^{\text{low}}\exp(-s_i)}{1 + \exp(-s_i)}. \tag{1.6}$$

It is critical to note that although the algorithm is applied in the log-transformation domain, i.e., in terms of the new control vector s throughout, we simply use a vector of control variables u instead of s to denote the control variable vector.

Chen et al. (2010) proposed using the gradient-projection method for production optimization problems with bound constraints. The gradient-projection method Nocedal and Wright (1999) handles the bound constraints in two steps. In the first step, we search for minimization along the steepest descent direction. When a bound is encountered, the search direction is "bent" or projected onto the bound constraint so that all points along the search path become feasible. The search path then becomes piecewise linear. The first local minimizer along this piecewise search direction is called the Cauchy point. The second step of the gradient-projection method is to search the face on which the Cauchy point is located for a new (improved) minimum. During this second step in optimization, the variables that form the active bound constraints at the Cauchy point are fixed and optimization is done in the space of the variables of the inactive bound constraints. Initially, in our work, we also attempted to use gradient projection to handle bound constraints, but we were unsuccessful. Apparently, the very approximate gradient generated with SPSA are not accurate enough to apply the gradient projection methodology reliably.

1.2.2 Control-only constraints and state-control constraints

Chen (2011) defined two types of constraints which normally need to dealt with in production optimization: linear control-only constraints and state-control constraints.

Linear control-only constraints consist solely of control variables, and are explicit linear functions of those control variables. The general idea of Rosen's gradient-projection method (Luenberger, 1984; Rao, 1965) is to project an unconstrained gradient onto the hyperplane of a linear equality constraint or a linear active inequality constraint, which ensures that any point on the search direction will satisfy the constraints. Zhang et al. (2010) and Forouzanfar et al. (2010) applied this method to handle a linear total injection rate constraint together with bound inequality constraints in their well-placement problem.

State-control constraints are nonlinear functions of the primary state variables and control variables. For example, the production liquid rate constraint is an implicit nonlinear function of the producer BHP controls. The most troublesome constraints are nonlinear state-control equality and inequality constraints. As the inequality constraints can be converted easily to equality constraints using active set method or slack variables (Nocedal and Wright, 1999), here we discuss only equality constraints. In the production optimization literature, two adjoint-gradient-based strategies have been applied to the nonlinear state-control equality of constraints:

generalized reduced gradient (GRG) (Zakirov et al., 1996; de Montleau et al., 2006) and approximate feasible direction (Sarma et al., 2008).

1.3 Introduction to Algorithms Applied in Production Optimization

In production optimization, we can use both gradient-free optimization algorithms and gradient-based optimization algorithms to obtain the optimal well control. Gradient-free algorithms, such as the finite-difference method, simultaneous perturbation stochastic approximation (SPSA) algorithm (Spall, 1998), ensemble-based optimization (EnOpt) algorithm (Chen et al., 2009), new unconstrained optimization algorithm (NEWUOA) (Powell, 2006), bound optimization by quadratic approximation (BOBYQA) (Powell, 2009), and quadratic interpolation model with an approximate gradient (QIM-AG) (Zhao et al., 2011), treat the reservoir simulator as a black box and evaluate the approximate gradient based on the output of the estimated objective values plus the input of perturbations of control variables. The finite-difference method requires evaluation of at least $n_u + 1$ objective function values for each optimization iteration and the computational cost is extremely expensive. SPSA is a "simplified" finite-difference method where all the parameters are perturbed at one time stochastically and the SPSA gradient is then calculated from one-sided or two-sided difference equation. Although the SPSA gradient is stochastic, its expectation is equal to the true gradient, and the search direction is always uphill (Spall, 1998, 2003) as the magnitude of the perturbation goes to zero. The SPSA algorithm has been applied in optimal well control (Wang et al., 2009; Zhao et al., 2011), optimal well placement (Bangerth et al., 2006) and history matching (Gao et al., 2007; Li and Reynolds, 2011).

EnOpt was first applied in the optimal well control problem in (Lorentzen et al., 2006) and then developed by Nwaozo (2006) and Chen et al. (2009). EnOpt requires generating an ensemble of control vectors and running the reservoir simulator for each of these control vectors in order to calculate the gradient from the cross-correlation between the control vectors and NPV's. In EnOpt, the controls for each well are usually assumed to be correlated in time, which leads to smooth optimal control settings.

NEWUOA and BOBYQA are quadratic model-based derivative-free algorithms proposed by Powell (2006, 2009). In NEWUOA, construction of the quadratic model is based on quadratic interpolation. The coefficients in the quadratic function are determined based on the quadratic function and the objective function being equal at a set of interpolation points. To promote computational efficiency in large-scale optimization problems, the number of interpolation points is usually much less than the number of coefficients in the quadratic model. The extra degrees of freedom are used to minimize the Frobenius norm of the difference between the term representing the approximate Hessian matrix in the quadratic model at the previous iteration and the Hessian in the updated quadratic model at the current iteration, i.e.,

$$\|G^{\ell} - G^{\ell-1}\|_F^2 = \sum_{i=1}^{n_u} \sum_{j=1}^{n_u} (G_{ij}^{\ell} - G_{ij}^{\ell-1})^2, \tag{1.7}$$

where $\| \cdot \|_F$ represents the Frobenius norm, G^{ℓ} is the approximate Hessian matrix at the ℓ^{th} iteration and the subscripts "i" and "j" refer to the entry in the i^{th} row and j^{th} column of the matrix. The updated quadratic model is then maximized using a trust-region method. This quadratic model is updated during iteration as more and better interpolation points become available. To optimize the objective function with simple bound constraints without derivatives, Powell (2009) proposed the BOBYQA algorithm. While NEWUOA builds an initial quadratic model based on at least $n_u + 2$ interpolation points before the optimization starts, BOBYQA needs the prescribed number of interpolation point to be greater than or equal to $2n_u + 1$ to build quadratic models. However, both NEWUOA and BOBYQA are very inefficient when the number of control variables is very large.

To overcome this limitation in NEWUOA and BOBYQA, Zhao et al. (2011) proposed the Quadratic Interpolation Model with Approximate Gradient (QIM-AG) algorithm based on a dynamic quadratic interpolation model for the maximization of the NPV. When the EnOpt preconditioned gradient is used, the algorithm is referred to as QIM-EnOpt, whereas when the SPSA gradient is used, the algorithm is referred to as QIM-SPSA. The QIM-AG algorithm uses all available evaluated points to construct dynamically a quadratic interpolation model along the iterative process. Both NEWUOA and QIM-AG build a quadratic approximate model based on a set of interpolation points and then seek the optimum for the quadratic model. While NEWUOA and BOBYQA construct the quadratic model by minimizing the Frobenious norm resulting from the difference between two approximate Hessian's in two consecutive iterations, the QIM-AG method constructs the quadratic model by minimizing the Frobenius norm of the approximate Hessian at the current iteration, subject to the constraint which the quadratic model is equal to the objective function evaluated at all the interpolation points.

Throughout the contemporary literature of production optimization, most of the gradient-free algorithms such as SPSA and QIM-AG are applied in order to handle the optimization problem using simple bound constraints. Chen et al. (2009) claim EnOpt is able to handle the problem with linear constraints as well as with simple bound constraints. In their application, the total water injection rate and the total production liquid rate are linear functions of control variables consisting of well injection rate and well production liquid rates. The total injection rate and total production liquid rate are truncated once they are violated, and then the total rate constraint is honored by reallocating proportionally the rates among wells according to the truncated values. Although this truncation technique seemed to give good results in their examples, it is not able to handle more complicated constraint types, e.g. state-control constraints. Dehdari and Oliver (2011) used

EnOpt to find the gradient and then used localization to improve the estimate of the gradient. To deal with the constraints, they used a method which eliminated non-negative constraints to decrease computation time, and an updating procedure to solve each iteration of SQP much faster than the base case. Shuai et al. (2011) applied multiscale regularization for both the EnOpt algorithm and the BOBYQA algorithm to estimate optimal well controls. After multiscale regularization, both methods obtained a NPV that equalled or exceeded the one from unregularized optimization. Wang et al. (2009) compared three different optimization algorithms: ensemble-based algorithm, SPSA algorithm, and the steepest ascent algorithm using finite-difference method and they concluded that the steepest ascent algorithm is the most efficient one and it gives reasonable results. In their study, an average of 10-20 SPSA gradients was used to generate the SPSA gradient, and this gradient was used as the search direction in the steepest ascent method. SPSA method resulted in the same NPV and well control variables obtained as when using the true gradient, but the SPSA method required far more reservoir simulation runs.

Gradient-based optimization algorithms require computation of the objective function's gradient with respect to control variables. As NPV and state-related constraints are implicit functions of the control variables, we are not able to calculate the gradient of the objective function explicitly. Sarma et al. (2008) applied the adjoint technique similar to the one used by Li et al. (2003) in history matching to calculate the gradient of NPV with respect to control variables and applied the adjoint gradient in the GRG algorithm and in the approximate feasible direction method. The adjoint gradient calculation with respect to well controls has significant computational advantages when the number of variables is large, as the number of adjoint runs does not depend on the number of variables. Therefore, it is suitable for problems with a large number of well controls to be adjusted, which is often the case when many wells and/or many sections of wells (smart wells) are involved with many control steps during the expected life the reservoir. Chen et al. (2010) used the augmented Lagrangian method to handle the nonlinear constraints, and the gradient-projection method to deal with bound constraints. They obtained excellent results, but they computed the gradient of the augmented Lagrangian function using the adjoint method which was implemented in an in-house limited application simulator. As mentioned earlier, commercial simulators do not generally have adjoint capability, and (to our knowledge) no commercial simulator incorporates implementation of the augmented Lagrangian method for production optimization. Our objective in this study is to develop implementation of the SPSA algorithm which can be used with the augmented Lagrangian method.

1.4 Research Objectives and Dissertation Outline

1.4.1 Research Objectives

The primary objective of our research is to develop practical optimization methods which can efficiently deal with large scale production optimization problems with bound, linear and nonlinear constraints. Specific elements in our project are:

1. To develop a code to approximate the gradient of the augmented Lagrangian function using the SPSA algorithm under various linear and nonlinear constraints on variables such as WOR, GOR, production and injection rates, and well-bore pressures.

2. To improve the SPSA algorithm using samples not only from a Bernoulli distribution but also from a Gaussian distribution in order to obtain a smoother approximate gradient.

3. To develop a method for estimating the SPSA parameters for both Bernoulli distribution and Gaussian distribution using experimental computation, and

to develop a combined search direction to improve the efficiency of the SPSA algorithm.

4. To develop a method for the scaling of different constraints for the SPSA algorithm, as well as to solve the tolerance convergence condition of the inner loop and the outer loop in the augmented Lagrangian algorithm.

5. To develop a theoretical comparison between several derivative-free algorithms in production optimization.

1.4.2 Dissertation Outline

There are five chapters. In Chapter 2, we present the SPSA algorithm using both a Bernoulli distribution and a Gaussian distribution, after which we present the experimental computation used to estimate the SPSA parameters in several case studies. In Chapter 3, we introduce the augmented Lagrangian method, including a discussion of methods for updating the Lagrangian multipliers and the penalty parameter. And then we use the augmented Lagrangian method with the SPSA gradient to solve constrained production optimization problems. In that chapter, we maximize the NPV subject to equality, and inequality constraints, as well as nonlinear constraints. In Chapter 4, we compare results obtained from the SPSA, EnOpt and simplex gradient methods for problems using simple bound constraints. Chapter 5 presents our conclusions and summarizes the research contributions of this study.

CHAPTER 2

COMPUTATIONAL EXPERIMENT ON SPSA PARAMETERS

The SPSA is a stochastic optimization algorithm, which obtains its stochastic gradient by simultaneously perturbing all components of the control vector in a stochastic way. Although the SPSA gradient is stochastic, it is always uphill for a sufficiently small perturbation size and its expectation is equal to the true gradient as the perturbation size goes to zero. We note that SPSA algorithm (Spall, 1992, 1998, 2003) provides an efficient alternative to the finite-difference method for estimating the gradient of an objective (cost) function when the number of optimization variables is large. The SPSA method and a derivative of SPSA based on Gaussian perturbations have been successfully applied in a variety of petroleum engineering applications (Bangerth et al., 2006; Li and Reynolds, 2011; Zhao et al., 2011).

This chapter introduces a general review about the SPSA algorithm developed by (Spall, 1992, 1998, 2003) where the symmetric ± 1 Bernoulli distribution is applied to generate an approximate gradient. We also present the modified SPSA proposed by Li and Reynolds (2011). The modified SPSA has the main desirable features similar to those of the SPSA, i.e., the approximate gradient generated by the modified SPSA algorithm gives an uphill direction for sufficiently small perturbation size, and the expectation of the modified SPSA gradient is equal to a smoothing covariance matrix times the true gradient with a bias in the approximation that goes to zero as the perturbation size goes to zero. Moreover, preconditioned steepest-ascent algorithm is proposed to estimate the optimal well controls when using the modified SPSA algorithm. The most important goal of this chapter is to investigate a general way to estimate SPSA parameters when we want to use the SPSA algorithm to maximize NPV of production. To solve the bound constraints, we apply the log-transformation to convert the optimization problem with bound constraints into one without bound constraints. Experimental computations on some specified cases are implemented to investigate a good choice of SPSA parameters when we want to maximize constrained production optimization problems on the production optimization step of closed- loop reservoir management.

2.1 The Basic SPSA Algorithm

For maximizing a general objective or cost function $O(u)$, the basic SPSA algorithm (Spall, 1992, 1998, 2003) updates the control vector using the SPSA gradient $\widehat{g_k}$ as the search direction,

$$u_{k+1} = u_k + a_k \widehat{g}_k, \tag{2.1}$$

where k is the iteration index and a_k is the step size. In production optimization, we denote the NPV, $J(u, y(u, m))$, simply by $J(u)$. Here $y = y(u, m)$ denotes the vector of primary variables solved for during the simulation run; y is a function of both the vector of all well controls u, as well as a function of the reservoir model m, which is assumed known or at least fixed during production optimization. The SPSA gradient at u_k is denoted by $\widehat{g_k}$ and can be obtained using a one-sided simultaneous perturbation,

$$\widehat{g}_k = \frac{J(u_k + c_k\Delta_k) - J(u_k)}{c_k}\Delta_k^{-1}, \tag{2.2}$$

or a two-sided simultaneous perturbation,

$$\widehat{g}_k = \frac{J(u_k + c_k\Delta_k) - J(u_k - c_k\Delta_k)}{2c_k}\Delta_k^{-1}, \tag{2.3}$$

where $c_k > 0$ is the perturbation size, $\Delta_k = [\Delta_{k1}, \Delta_{k2}, ..., \Delta_{kn}]^T$ is a random perturbation vector and its inverse is defined as an element-wise inverse, i.e., $\Delta_k^{-1} =$

$[\Delta_{k1}^{-1}, \Delta_{k2}^{-1}, ..., \Delta_{kn}^{-1}]^T$. It is important to note that here u_k denotes the vector u at the k^{th} iteration of the optimization algorithm, not the k^{th} component of u. The common choice for the random vector Δ_k is a sample from the symmetric ± 1 Bernoulli distribution, i.e., Δ_{ki} can only takes values of 1 or -1. In this case, by definition, Δ_k^{-1} is equal to Δ_k. Here, the SPSA algorithm is denoted as B-SPSA when the Bernoulli distribution is used to generate the SPSA gradient. It is shown (Spall, 2003) that the expectation of the SPSA gradient is the true gradient and the SPSA gradient is always an uphill direction as c_k goes to zero.

2.2 SPSA Gradient Using a Gaussian Perturbation

2.2.1 Smoothed stochastic search direction

Li and Reynolds (2011) suggested calculating an SPSA gradient using a Gaussian perturbation vector sampled from a normal distribution with mean equal to the dimensional zero vector and covariance matrix, C_U, i.e, $\delta u_k \sim N(0, C_U)$. Using this procedure gives a one-sided simultaneous perturbation,

$$\widehat{g}_k = \frac{J(u_k + c_k \delta u_k) - J(u_k)}{c_k} \delta u_k, \tag{2.4}$$

and a two-side simultaneous perturbation,

$$\widehat{g}_k = \frac{J(u_k + c_k \delta u_k) - J(u_k - c_k \delta u_k)}{2c_k} \delta u_k. \tag{2.5}$$

We can use either one-sided or two-sided approximation of Eqs. 2.4 and 2.5 to compute a stochastic gradient. The random vector, δu_k, is calculated by

$$\delta u_k = C_U^{1/2} Z_k, \tag{2.6}$$

where Z_k is an n_u-dimensional column vector and its components are independent standard random normal deviates, $C_U^{1/2}$ is a "square root" of the prior covariance matrix C_U. In our computations, we use $C_U^{1/2} = L$ where L is the lower triangular matrix from the Cholesky decomposition of C_U, i.e., $C_U = LL^T$. The "steepest ascent algorithm" with the search direction calculated with Eqs. 2.4 and 2.5 is denoted by G-SPSA where the G refers to Gaussian. Although using a perturbation δu_k from the normal distribution $N(0, C_U)$ does not yield an SPSA gradient in the sense of Spall (1992), Li and Reynolds (2011) show that the stochastic gradient of Eq. 2.4 gives an uphill direction for sufficiently small c_k and that when $\widehat{g}_k$ is given by Eq. 2.4, its expectation is

$$E[\widehat{g}_k] = C_U \nabla O(u_k) + O(c_k), \tag{2.7}$$

and when $\widehat{g}_k$ is the random vector defined by Eq. 2.5, the expectation of $\widehat{g}_k$ is given by

$$E[\widehat{g}_k] = C_U \nabla O(u_k) + O(c_k^2), \tag{2.8}$$

assuming that the objective function $O(u)$ is three time continuously differentiable. Thus, the "modified SPSA algorithm" of Li and Reynolds (2011) shares important characteristics of the SPSA algorithm of Spall (1992, 1998, 2003). Moreover, for history matching problems, Li and Reynolds (2011) found that generating $\widehat{g}_k$ from Eq. 2.4 (or Eq. 2.5) results in a more robust optimization algorithm than the one obtained by calculating $\widehat{g}_k$ from Eq. 2.2 (or Eq. 2.3) using a perturbation sampled from the Bernoulli distribution and multiplied by C_U and using $-C_U\widehat{g}_k$ as the search direction. Li and Reynolds (2011) were concerned with minimizing rather than maximizing an objective function and in their history matching application, a covariance from the prior model is always available. However, in the optimal well control problem of interest here, there is no natural prior model for the vector of well controls u. Thus, we are effectively creating an ad hoc covariance which serves to generate a smoothed stochastic search direction vector $\widehat{g}_k$ from Eq. 2.4 with expectation

$C_U \nabla O(u_k)$.

2.2.2 Covariance matrix in the G-SPSA algorithm

We propose the algorithm

$$u_{k+1} = u_k + \widehat{g}_k \tag{2.9}$$

with $\widehat{g}_k$ computed from Eq. 2.4. Taking the expectation of Eq. 2.9 and using Eq. 2.7, it follows that as c_k goes to zero

$$E[u_{k+1}] = E[u_k] + C_U \nabla O(u_k), \tag{2.10}$$

where $E[\cdot]$ denotes expectation. Thus Eq. 2.9 is similar to a smoothed or preconditioned steepest ascent algorithm. Thus the resulting vector of controls are expected to be smoother when they are generated with G-SPSA than with an SPSA using a Bernoulli distribution assuming that C_U is generated from a covariance function that has a correlation length with length greater than or equal to two control steps. Although it is not desirable to have controls at different wells correlated, we may wish to have the well controls at an individual well vary smoothly with time. This may be desirable from a practical view; the operator does not want short term fluxuations in the well controls (rate or pressure). From a theoretical viewpoint, using C_U can provide regularization that could possibly lead to a more robust algorithm. There is however a caveat. If we use a covariance matrix C_U with a long correlation length, it will be difficult to produce a bang-bang optimal solution where, for example, the optimal water injection rate can be the maximum allowable rate at one control step and zero at the next control step, (Zandvliet et al., 2007; Wang et al., 2009). Nevertheless, the examples shown here are based on using a covariance matrix C_U to force some degree of smoothness on the estimated optimal well controls on a well by well basis, i.e., C_U is a block diagonal matrix of the form

$$C_U = \begin{bmatrix} C_{U^1} & 0 & 0 & 0 \\ 0 & C_{U^2} & 0 & 0 \\ . & . & . & . \\ 0 & 0 & 0 & C_{U^{n_w}} \end{bmatrix}, \tag{2.11}$$

where n_w is the number of wells and $C_{U^\ell}, \ell = 1, 2, ...n_w$, is the covariance matrix used to force some degree of smoothness on the well controls for well ℓ. We let u^ℓ denote the vector of well controls at well ℓ, $u^\ell = [u_1^\ell, u_2^\ell, ..., u_{N_c}^\ell]^T$ where u_i^ℓ denotes the well control (rate or pressure) for the i^{th} control step at well ℓ. Note with this notation, the complete control vector u has the form

$$u = \begin{bmatrix} u^1 \\ u^2 \\ . \\ . \\ u^{n_w} \end{bmatrix}, \tag{2.12}$$

and

$$C_U u = \begin{bmatrix} C_{U^1} u^1 \\ C_{U^2} u^2 \\ . \\ . \\ C_{U^{n_w}} u^{n_w} \end{bmatrix}. \tag{2.13}$$

With the gradient ordered the same way as u in Eq. 2.12,

$$C_U \nabla_u O(u^k) = \begin{bmatrix} C_{U^1} \nabla_{u^1} O(u^k) \\ C_{U^2} \nabla_{u^2} O(u^k) \\ \cdot \\ \cdot \\ C_{U^{nw}} \nabla_{u^{nw}} O(u^k) \end{bmatrix}, \tag{2.14}$$

i.e., multiplying the gradient $\nabla_{u^\ell} O(u^k)$ by C_{U^ℓ} effectively smooths u^ℓ. Denoting the (i, j) entry of C_{U^ℓ} by C^ℓ_{ij}, smoothing based on the spherical model is given by

$$C^\ell_{i,j} = \begin{cases} \sigma_\ell^2 \left[1 - \frac{3}{2} \left(\frac{|i-j|}{N^\ell_s} \right) + \frac{1}{2} \left(\frac{|i-j|}{N^\ell_s} \right)^3 \right], & \text{if } |i-j| < N^\ell_s, \\ 0, & \text{otherwise.} \end{cases} \tag{2.15}$$

Here, i and j respectively refer to control step i and control step j and N^ℓ_s is the number of control steps over which we wish the control at well ℓ to be correlated. Finally, σ_ℓ is the standard deviation in the control of well ℓ. Note that C_U not only promotes smoothness of the controls but also is used to generate the perturbation for calculating the stochastic gradient. Because of this, we set σ_ℓ equal to about 2 percent of the maximum range for the well control (original control) used at the particular well. For example, if producing well ℓ is controlled by bottom hole pressure, $p_{wf,\ell}$ and we require that $p_{wf,\ell}$ satisfy the bound

$$0 \leq a \leq p_{wf,\ell} \leq b, \tag{2.16}$$

at all control steps, then $\sigma_\ell = 0.02[b - a]$ is appropriate. This recommendation is for the case where we do not transform to the log-transform domain. How to choose σ_ℓ for all ℓ when we transform to the log-transform domain using Eq. 1.5 will be discussed more detail later. In the log-transform domain, the covariance should properly be denoted by C_S but as mentioned previously, even when working in the log-transform domain, we still denote the optimization vector by u and the covariance matrix by C_U. For all examples presented here, the optimization is done in terms of the transformed variables (Eq. 1.5) because via many experimental computations we found that this procedure yields a higher NPV than is obtained by enforcing bounds by truncation. In truncation, whenever, a component of u_{k+1} computed from Eq. 2.21 is greater than its upper bound we set that component equal to its upper bound and whenever a component of u_{k+1} is less than its specified bound, we set that component equal to its lower bound.

2.3 Choice of The Stepsize a_k and Perturbationsize c_k

As the expectation of the stochastic gradient is the true gradient (or C_U times the true gradient), we expect to obtain a better approximation of the desired search direction by generating M stochastic gradients from Eq. 2.2 (or Eq. 2.4) by using M different Δ_k's (or δu_k's) and then computing an average stochastic gradient $\overline{\widehat{g}}_k$ by

$$\overline{\widehat{g}}_k(u_k) = \frac{1}{M}\sum_{j=1}^{M} \widehat{g}_{k,j}(u_k). \tag{2.17}$$

Then we replace Eq. 2.1 by

$$u_{k+1} = u_k + a_k \overline{\widehat{g}}_k. \tag{2.18}$$

According to Spall (1998, 2003), the step size a_k and perturbation size c_k are defined by the following equations:

$$a_k = \frac{a}{(k+A+1)^{\alpha}}, \tag{2.19}$$

$$c_k = \frac{c}{(k+1)^{\gamma}}, \tag{2.20}$$

where a, A, c, α and γ are positive real numbers which satisfy $A \geq 0, \alpha - 2\gamma > 0$ and

$3\gamma - \alpha/2 > 0$. The choice of these parameters can have a fairly significant effect on the performance of the SPSA algorithm (Spall, 2003).

We find it easier to specify valves of a, A and c if we first scale the stochastic gradient. Specifically, we normalize the search direction by its infinity norm so that

$$u_{k+1} = u_k + a_k \frac{\overline{\widehat{g}}_k}{\|\overline{\widehat{g}}_k\|_\infty} . \tag{2.21}$$

With such a normalization, we have a better idea of how to choose an initial value of A and a, and then a_k so that we can potentially vary the components of u_{k+1} over its expected range. Our guideline for choosing parameters is discussed in the following steps:

- In all cases, we set $\alpha = 0.602$ and $\gamma = 0.101$, these choices result from a theoretically based recommendation of Spall (1998).
- To choose A, we first set the maximum number of allowable iterations, $k_{\max}$. Then, we choose a value of A equal to 10% of the maximum number of allowable iterations. As discussed later, choosing $k_{\max}$ should be based on the total number of control variables in the specific problem. If there are sufficient computation resource available, using $k_{\max}$ greater than or equal to the number of control variable is recommended.
- Next we choose a_0 equal to the maximum change in any component of u we allow at the first iteration. Typically, this could be on the order of 1/2 the distance to the nearest bound. But since we work in term of the log-transform variables, we suggest the initial stepsize in the range from 1.0 to 3.0 works fine.
- After choosing A and a_0, one can choose a such that at $k = 0$, a satisfies $a_0 = a/(1+A)^\alpha$, i.e., $a = a_0(1+A)^\alpha$.

- To estimate the value of c in Eq. 2.20 after setting the maximum number of allowable iterations, $k_{\max}$, we let the minimum allowable change in the perturbation size be denoted by $c_{\min}$, then calculate c_0 from $c_{\min} = c_0/(k_{\max}+1)^{\gamma}$. When optimizing in the log-transform domain, we have found via experiment that $c_{\min}$ from 0.01 to 0.1 is appropriate, assuming a Bernoulli perturbation is used or covariance matrices C_U (Eq. 2.15) have variance on the order of 1.0.

If we work in the original domain instead of the log-transform domain, one can scale each variable u_i of the overall control vector u by defining

$$\widehat{u}_i = \frac{u_i - u_i^{\text{low}}}{u_i^{\text{up}} - u_i^{\text{low}}},\ i = 1, 2, ..., n_{\text{u}}, \tag{2.22}$$

and then the bounds on all $\widehat{u}_i$'s become 0 and 1, i.e.,

$$0 \leq \widehat{u}_i \leq 1. \tag{2.23}$$

Assuming with this rescaling, we optimize using $\widehat{u}$ instead of u as the vector of optimization variables (transformed well controls) and then $c_{\min}$ from 0.01 to 0.1 is still appropriate. If working in terms of the original optimization variables u and using Bernoulli perturbation, we see that in order for the calculation of $\widehat{g}_k$ from Eq. 2.2 to be meaningful, we must have the calculation $\delta J_k = J(u_k + c_k\Delta_k) - J(u_k)$ to be meaningful, i.e., if we can calculate the two values of J accurate to 6 digits, and we generate a perturbation size c_k so small that $J(u_k + c_k\Delta_k)$ and $J(u_k)$ agree to six or more digits, then δJ_k will have no correct digits and $\widehat{g}_k$ computed from Eq. 2.2 will just reflect noise, i.e., round off error. To obtain a meaningful $\widehat{g}_k$, we must ensure all $c_k \geq c_{\min}$ where $c_{\min}$ is chosen so that $J(u_k + c_{\min}\Delta_k) - J(u_k)$ has at least a couple of correct digits. To ensure this happens for every optimal well control, the optimization problem could require costly computational experiments. Fortunately, we have found that a value of $c_{\min}$ between 0.01 and 0.1 works reasonably well when we work in the log-transform domain or in term of $\widehat{u}$, see Eq. 2.22.

2.4 Combined Search Direction SPSA Algorithm

Due to the influence of noise and the stochastic nature of the gradient approximation, $\widehat{g}_k$ or $\overline{\widehat{g}}_k$ may not always be the best iterative direction, especially if the function to be maximized has long narrow valleys. In such a case, the conjugate gradient (CG) method often works better than the steepest ascent method in deterministic optimization (Schraudolh and Graepel, 2002). Based on this idea, Xu (2010) proposed a combined direction stochastic approximation algorithm. At each iteration of this algorithm, a weighted combination of the current approximate gradient and some former gradient is chosen as the direction. This method may work worse than steepest ascent at a specified iteration. However, it may work better overall, if a "good" former gradient can be chosen. A good gradient at iteration $k-1$ is defined as one which at iteration k gives a gradient close to zero because if we calculated the true gradient, at a maximum of the cost function, the gradient would be zero. Xu (2010) recommended the following procedure:

$$u_{k+1} = u_k + a_k \widehat{p}_k, \tag{2.24}$$

where

$$\widehat{p}_k = \widehat{g}_k + \frac{\widehat{g}_k^T \widehat{g}_{f(k)}}{\|\widehat{g}_{f(k)}\|^2} \widehat{g}_{f(k)}. \tag{2.25}$$

Here, $f(k)$ is an integer subscript which satisfies $f(k) \leq k-1$; a_k is still computed from Eq. 2.19 using the guideline for the parameters that we presented previously. The choice of $f(k)$ is the key point of this method. If the norm of the approximate gradient is close to zero at some u_t where t is the iteration index, then the gradient or search direction at u_{t-1} is a "good" direction at least at that iteration, i.e., it is reasonable to believe that the direction $\widehat{g}_{t-1}$ is a good ascent direction. Therefore,

$\widehat{g}_{t-1}$ can be used to correct the current approximate gradient $\widehat{g}_k$. In this case, the procedure to choose $f(k) = t-1$ is

$$f(k) = \begin{cases} (\arg\min \|\widehat{g}_i\|) - 1 \quad \text{for } k-r+1 \le i \le k, & \text{if } k > r, \\ (\arg\min \|\widehat{g}_i\|) - 1 \quad \text{for } 1 \le i \le k, & \text{if } k < r, \end{cases} \tag{2.26}$$

where r is a positive integer specified by the user. In our example, we use $r = 10$. We refer to this method as ComSPSA1. Note that $\widehat{p}_k$ is a linear combination of $\widehat{g}_k$, the normal SPSA or GSPSA search direction and a search direction $\widehat{g}_{f(k)}$ that was a good search direction at a previous iteration. Also note that Eq. 2.25 can be rewritten as

$$\widehat{p}_k = \widehat{g}_k + \cos\theta \frac{\|\widehat{g}_k\|}{\|\widehat{g}_{f(k)}\|} \widehat{g}_{f(k)}, \tag{2.27}$$

where θ is the angle between $\widehat{g}_k$ and $\widehat{g}_{f(k)}$. Note the norm of the vector

$$\nu \equiv \cos\theta \frac{\|\widehat{g}_k\|}{\|\widehat{g}_{f(k)}\|} \widehat{g}_{f(k)}, \tag{2.28}$$

added to $\widehat{g}_k$ to form $\widehat{p}_k$ satisfies $\|\nu\| \le \|\widehat{g}_k\|$.

Using this procedure to combine search directions, we cannot guarantee the combined search direction $\widehat{p}_k$ is always uphill at every iteration. Recall that the SPSA gradient is always an uphill direction if we use either the Bernoulli distribution (Spall, 2003) or the Gaussian distribution (Li and Reynolds, 2011) to generate the SPSA gradient and c_k is sufficiently small. Furthermore, if the current gradient is in the opposite direction of a "good" former gradient $\widehat{g}_{f(k)}$, the current combined search direction would be equal to zero. However, Xu (2010) believed that this algorithm may work better over several iterations. Xu (2010) also proved that under standard assumptions (Fabian, 1968; Bertsekas and Tsitsiklis, 2003) either $O(u_k) \to -\infty$ or $O(u_k)$ converges to a finite value and $\lim_{k\to+\infty} \nabla O(u_k) = 0$ with probability 1. In this study, we propose a new way to calculate a search direction by combining all previous SPSA gradients that have an angle relative to the current stochastic gradient of less than ±90 degrees. This is similar to the ComSPSA1 algorithm in that we cannot guarantee that the new search direction is uphill at each iteration. With this procedure (referred to as ComSPSA2), the search direction is defined as

$$\widehat{p}_k = \begin{cases} \widehat{g}_k + \displaystyle\sum_{j=k-r+1}^{k-1} \max\left(0, \frac{\widehat{g}_k^T \widehat{g}_j}{\|\widehat{g}_j\|^2}\right)\widehat{g}_j & \text{if } k > r, \\ \widehat{g}_k + \displaystyle\sum_{j=1}^{k-1} \max\left(0, \frac{\widehat{g}_k^T \widehat{g}_j}{\|\widehat{g}_j\|^2}\right)\widehat{g}_j & \text{if } 1 < k \le r, \end{cases} \tag{2.29}$$

where r is a given positive integer, $r = 10$ in the examples considered later.

Because $\widehat{g}_k^T \widehat{g}_j = \|\widehat{g}_k\| \|\widehat{g}_j\| \cos\theta$ where θ is the angle between $\widehat{g}_k$ and $\widehat{g}_j$, the vector $\widehat{g}_j$ is multiplied by zero unless $-\pi/2 < \theta < \pi/2$. Eqs. 2.25 and 2.26 are also considered when all the $\widehat{g}_k$'s are replaced by $\overline{\widehat{g}}_k$'s using Eq. 2.17.

2.5 Computational Results

2.5.1 Production Optimization with simple bound constraints for a three-channel reservoir

In this example, we implement production optimization for a horizontal reservoir which has a uniform grid system, $25 \times 25 \times 1$ with $\Delta x = \Delta y = 100$ ft (Zhao et al., 2011). The thickness of the reservoir is 20 ft. We consider only the two-phase flow of water and oil. The porosity is homogeneous. The log-permeability distribution is shown in Fig. 2.1 with high permeability channels. The initial reservoir pressure is 3800 psi and the initial water saturation is 0.2. There are a total of 13 vertical wells with 4 production wells and 9 injection wells arranged in a five-spot well pattern as shown in Fig. 2.1. We set each injection well under water injection rate control with a lower bound of 0 STB/D and an upper bound of 2000 STB/D, and each production well under bottom hole pressure (BHP) control with a lower bound of 1500 psi and an upper bound of 6000 psi. The anticipated total project life is 1800 days and the control step size is set equal to 180 days so we have 10 control steps. The total number of control variables is $(4+9) \times 10 = 130$. To optimize NPV, the oil price is set at \$50/STB, the water injection rate cost at \$0/BBL, the water production cost at \$5.56/STB, and the annual discount rate is 0%.

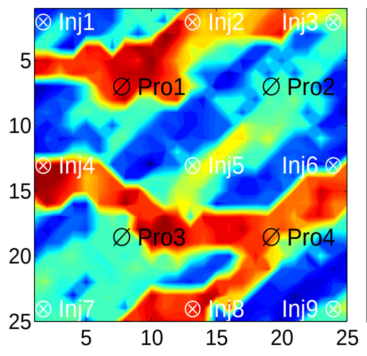


Figure 2.1: ln(k) distribution.

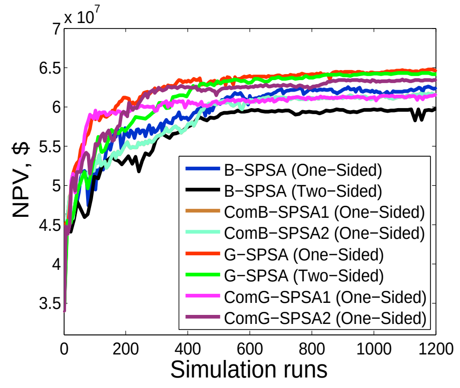


Figure 2.2: NPV versus the number of simulation runs.

The initial guess for water injection rate control of each injection well is set to 300 STB/D, and the initial guess for BHP control of each production well is set to 3500 psi. To deal with the bound constraints, we use the log-transformation to convert the bound control variables into the unbounded control variables. The constrained problem turn into an unconstrained problem. The condition for terminating the algorithm is based on the maximum number of allowable iterations.

To compare the SPSA algorithm's performance resulting from a Bernoulli perturbation and a Gaussian perturbation, we use the average of five SPSA gradients (Eq. 2.17) calculated by one-sided simultaneous perturbation and two-sided simultaneous perturbation (Eqs. 2.2, 2.3, 2.4 and 2.5). The normalized search direction (Eq. 2.21) is applied to update the control vector. In the G-SPSA algorithms, we use a variance pair of (1,1), of which the first number corresponds to the transformed variable of BHP control and the second number corresponds to the transformed variable of water injection rate control, and a correlation length of $N_s = 10$ to generate the covariance matrix C_U (Eq. 2.15). The value of α and γ in these equations is similar to the value suggested by Spall (1998), $\alpha = 0.602$ and $\gamma = 0.101$. By setting the maximum number of allowable iterations, $k_{\max}$, equal to 200 which is almost two times the number of control variables (130), then we obtain the value of $A = 20$. Since we compute the search direction as an average of five SPSA gradients, $k_{\max} = 200$ corresponds to 1200 simulation runs. By setting the value of a_0 equal to 2.0, we obtain the value of $a = 13.0$. We set $c_{\min}$ equal to 0.0585 which gives $c = 0.1$. We will discuss in more detail a procedure for choosing the SPSA parameters using experimental computations in the later parts of this example.

Table 2.1: The performance of different algorithms from one optimization of each algorithm with one initial random seed ($M = 5$, $k_{max} = 200$, $A = 20$, $a_0 = 2.0$, and $c_0 = 0.1$; $N_s = 10$ and $\sigma^2 = (1, 1)$).

Algorithm	No. of simulations	Final NPV,$\times 10^7$\$
B-SPSA(One-Sided)	1200	6.235
B-SPSA(Two-Sided)	1200	5.976
ComB-SPSA1(One-Sided)	1200	6.154
ComB-SPSA2(One-Sided)	1200	6.024
G-SPSA(One-Sided)	1200	6.462
G-SPSA(Two-Sided)	1200	6.413
ComG-SPSA1(One-Sided)	1200	6.153
ComG-SPSA2(One-Sided)	1200	6.337

The comparison of the performance of the SPSA algorithms is presented in Fig. 2.2 and Table 2.1. Note that all values of NPV are compared at the 1200^{th} simulation run. The G-SPSA algorithm using one-sided simultaneous perturbation gives the highest final NPV \$$6.462 \times 10^7$ which is about 0.7% greater than that of G-SPSA algorithm using two-sided simultaneous perturbation, and is about 3.6% greater than that of B-SPSA algorithm using one-sided simultaneous perturbation, and 8.0% greater than that of B-SPSA algorithm using two-sided simultaneous per-

Table 2.2: Average NPV obtained from five runs of each algorithm with five different initial seeds ($M = 5$, $k_{\max} = 200$, $a_0 = 2.0$, and $c_0 = 0.1$; $N_s = 10$ and $\sigma^2 = (1, 1)$).

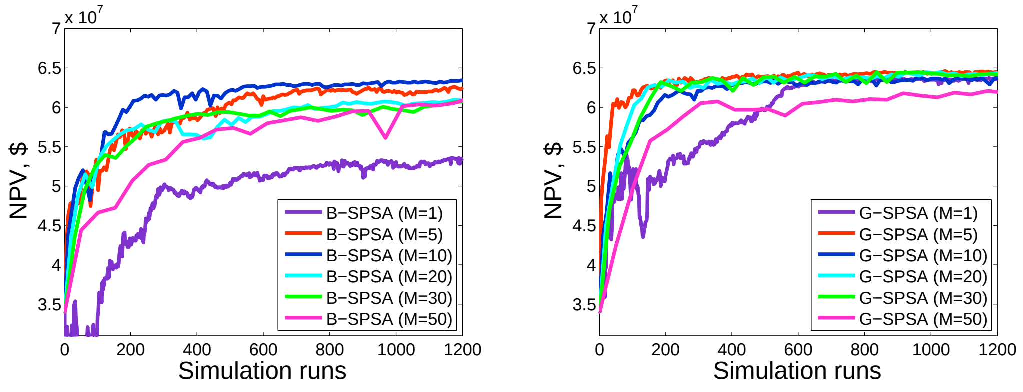
Algorithm	No. of simulations	Final NPV, $\times 10^7$\$
B-SPSA(One-Sided)	1200	6.249
B-SPSA(Two-Sided)	1200	6.006
ComB-SPSA1(One-Sided)	1200	6.152
ComB-SPSA2(One-Sided)	1200	6.094
G-SPSA(One-Sided)	1200	6.433
G-SPSA(Two-Sided)	1200	6.408
ComG-SPSA1(One-Sided)	1200	6.387
ComG-SPSA2(One-Sided)	1200	6.407

turbation. The most important thing to note is that the NPV from any SPSA algorithms using a one-sided simultaneous perturbation is always greater than that obtained from the corresponding two-sided simultaneous perturbation. Also the NPV of the G-SPSA algorithm increases faster than that of the B-SPSA algorithm in the sense that G-SPSA achieves a value close to its final maximum value in fewer iterations. Although we expect that the combined search direction based on conjugate gradient can result in a better NPV, the results in Table 2.1 show that the NPV of combined search direction algorithm is always less than that of the corresponding SPSA algorithm.

Table 2.3: Average NPV obtained from five runs of each algorithm with five different initial seeds when we change the value of $k_{\max}$ at fixed number of simulation runs, 1200 ($M = 5$, $a_0 = 2.0$, and $c_0 = 0.1$; $N_s = 10$ and $\sigma^2 = (1, 1)$).

Algorithm	Simulations	$k_{\max}$	A	Final NPV, $\times 10^7$\$
B-SPSA(One-Sided)	1200	$k_{\max} = 200$	$A = 20$	6.249
B-SPSA(Two-Sided)	1200	$k_{\max} = 110$	$A = 11$	5.982
G-SPSA(One-Sided)	1200	$k_{\max} = 200$	$A = 20$	6.433
G-SPSA(Two-Sided)	1200	$k_{\max} = 110$	$A = 11$	6.434

As all the algorithms have a stochastic component, drawing conclusions based on the results of one specific case may not be meaningful. Thus, we applied each of the stochastic algorithms with five different initial random seeds for generating the first stochastic perturbation. The computational results for these five runs are summarized in Table 2.2. The results shown in Table 2.2 represent the average NPV obtained from five runs of each stochastic algorithm with five different initial random seeds. From Table 2.2, we see that the conclusions drawing from one optimization run with one initial random seed apply for the average result of different seeds, i.e., the SPSA algorithm using Gaussian perturbations gives the best NPV and one-sided simultaneous perturbation is more computationally efficient than two-sided simultaneous perturbation, i.e., for a fixed number of iteration one-sided simultaneous perturbations yields a higher NPV. Moreover, the two ComG-SPSA algorithms did not prove superior to the Li-Reynolds G-SPSA method so the ComG-SPSA algorithms will not be discussed further. To make the above conclusion more strong, we run the SPSA algorithm using two-sided simultaneous perturbation with another maximum number of allowable iterations, $k_{\max} = 110$, which corresponds to the fixed number of simulation runs equal to 1200 and the number of SPSA gradients equal to 5 ($M = 5$). Because when we use the value of $M = 5$, at each iteration, the SPSA algorithm using two-sided simultaneous perturbation needs $(2 \times M + 1) = 11$ reservoir simulation runs to generate a steepest-ascent direction, and to obtain an updated NPV value. With $k_{\max} = 110$, we obtain the value of $A = 11$. Note that we still keep the initial step size and perturbation size constant, i.e., $a_0 = 2.0$ and $c_0 = 0.1$. The average NPV's obtained from five runs of each algorithm with five different initial seeds in Table 2.3 show that the B-SPSA algorithm with one-sided simultaneous perturbation works better than that using two-sided simultaneous perturbation. Meanwhile, for the G-SPSA algorithm, the NPV obtained using two-sided simultaneous perturbation is slightly greater than is obtained with one-sided simulta-



(a) B-SPSA (b) G-SPSA

Figure 2.3: NPV versus the number of simulation runs with different values of M ($k_{\max} = 200$, $A = 20$, $a_0 = 2.0$, and $c_0 = 0.1$; $N_s = 10$ and $\sigma^2 = (1, 1)$; the results obtained from one run of each algorithm wit one initial seed).

neous perturbation. From the preceding comparisons, we can find that the G-SPSA and B-SPSA algorithms using one-sided simultaneous perturbation is more computationally efficient than those using two-sided simultaneous perturbation, so from this point on we focus on the algorithms using one-sided simultaneous perturbation.

Fig. 2.3 presents NPV versus the number of simulation runs of the B-SPSA and G-SPSA algorithms as the number of B-SPSA gradients or G-SPSA gradients used to generate the average stochastic gradient $\overline{\widehat{g}}_k$ (see Eq. 2.17) changes. Except for changing the number of SPSA gradients which are averaged to compute the average stochastic gradient $\overline{\widehat{g}}_k$ (see Eq. 2.17), all other parameters are kept the same, i.e., $A = 20$, $a_0 = 2.0$ and $c = 0.1$. The normalized search direction (Eq. 2.21) is used to update control variables. When the number of SPSA gradient changes, the total number of simulation runs required to compute $\overline{\widehat{g}}_k$ changes. To be consistent when comparing the results, the NPV of each case will be compared at the 1200^{th} simulation runs, so the total computational work of each algorithm is the same. When the number of SPSA gradients is set equal to 1, the obtained NPV is always the worst. As we increase the value of M, the NPV becomes better. The value of $M = 5$ or $M = 10$ gives better NPV results than other values of M when we use

Table 2.4: Average NPV obtained from five runs of each algorithm with five different initial random seeds when we change the value of M ($k_{\max} = 200$, $A = 20$, $a_0 = 2.0$, $c_0 = 0.1$; $N_s = 10$ and $\sigma^2 = (1, 1)$).

Algorithm	No. of simulations	Final NPV,$\times 10^7$\$
B-SPSA ($M = 1$)	1200	5.380
B-SPSA ($M = 5$)	1200	6.249
B-SPSA ($M = 10$)	1200	6.184
B-SPSA ($M = 20$)	1200	6.131
B-SPSA ($M = 30$)	1200	6.167
B-SPSA ($M = 50$)	1200	6.111
G-SPSA ($M = 1$)	1200	6.256
G-SPSA ($M = 5$)	1200	6.433
G-SPSA ($M = 10$)	1200	6.454
G-SPSA ($M = 20$)	1200	6.316
G-SPSA ($M = 30$)	1200	6.348
G-SPSA ($M = 50$)	1200	6.198

both Bernoulli distribution and Gaussian distribution to perturb control variables. However, increasing M does not necessarily mean that NPV will increase. For the value of M is greater than or equal to 20, the NPV obtained is less than the NPV obtained with $M = 5$ or $M = 10$, especially for the Bernoulli distribution case.

The results shown in Table 2.4 for each stochastic optimization algorithm represent the average NPV obtained from the five optimization runs of each algorithm with five different initial random seeds when we change the value of M. Note that the same initial random seed is used for each optimization algorithm. For the B-SPSA algorithm, the value of $M = 5$ results in the best NPV of \$$6.249 \times 10^7$ which is about 16% greater than that obtained with $M = 1$, and is about 2.3% greater than is obtained with $M = 50$. For the G-SPSA algorithm, the value of $M = 10$ leads to the best NPV of \$$6.454 \times 10^7$ which is about 3.1% greater than the NPV obtained with $M = 1$, and is about 4.1% greater than the NPV obtained with $M = 50$, and is slightly better than the NPV obtained with $M = 5$. Again, we fix the total number of simulation runs at 1200, corresponding to each value of M, we obtain

Table 2.5: Average NPV obtained from five runs of each algorithm with five different initial random seeds when we change the value of M and the corresponding value of $k_{\max}$ at fixed number of simulation runs, 1200 ($a_0 = 2.0$, $c_0 = 0.1$; $N_s = 10$ and $\sigma^2 = (1, 1)$).

Algorithm	Simulations	$k_{\max}$	A	Final NPV,$\times 10^7$\$
B-SPSA ($M = 1$)	1200	$k_{\max} = 600$	$A = 60$	5.267
B-SPSA ($M = 5$)	1200	$k_{\max} = 200$	$A = 20$	6.249
B-SPSA ($M = 10$)	1200	$k_{\max} = 110$	$A = 11$	6.249
B-SPSA ($M = 20$)	1200	$k_{\max} = 57$	$A = 5$	6.231
B-SPSA ($M = 30$)	1200	$k_{\max} = 39$	$A = 3$	6.163
B-SPSA ($M = 50$)	1200	$k_{\max} = 24$	$A = 2$	6.083
G-SPSA ($M = 1$)	1200	$k_{\max} = 600$	$A = 60$	6.238
G-SPSA ($M = 5$)	1200	$k_{\max} = 200$	$A = 20$	6.433
G-SPSA ($M = 10$)	1200	$k_{\max} = 110$	$A = 11$	6.416
G-SPSA ($M = 20$)	1200	$k_{\max} = 57$	$A = 5$	6.325
G-SPSA ($M = 30$)	1200	$k_{\max} = 39$	$A = 3$	6.395
G-SPSA ($M = 50$)	1200	$k_{\max} = 24$	$A = 2$	6.221

one value of $k_{\max}$ which is used to estimate the value of A in Eq. 2.19. Note that we use the SPSA algorithm with one-sided simultaneous perturbations, therefore, at each iteration we need $M + 1$ reservoir simulation runs to generate a steepest-ascent direction, and to obtain an updated NPV value. Corresponding to each value of M, we obtain a maximum number of allowable iterations by roughly using the following equation: $k_{\max} = 1200/(M + 1)$ (see Table 2.5). We again keep the initial step size and perturbation size constant, i.e., $a_0 = 2.0$ and $c_0 = 0.1$. Fig. 2.4 presents NPV versus the number of simulation runs of the G-SPSA algorithm as the value of M used to generate the average stochastic gradient $\overline{\overline{g}}_k$ (see Eq. 2.17) changes and the corresponding $k_{\max}$ changes. Note that Fig. 2.4 present the results obtained from one optimization run with one initial random seed. The average NPV 's obtained from five runs of each algorithm with five different initial seeds are presented in Table 2.5. The results from Fig. 2.4 and Table 2.5 show that the value of $M = 5$ or $M = 10$ gives better NPV results after 1200 reservoir simulation runs than other values of M

when we use the corresponding maximum number of allowable iterations.

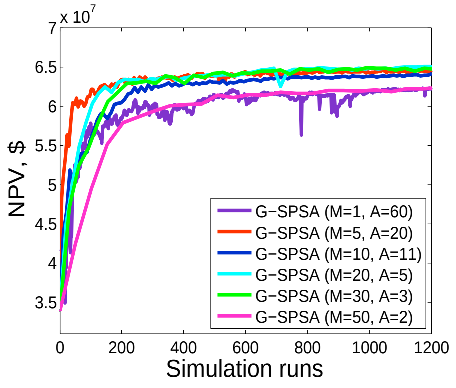


Figure 2.4: NPV versus the number of simulation runs with different values of M and $k_{\max}$.

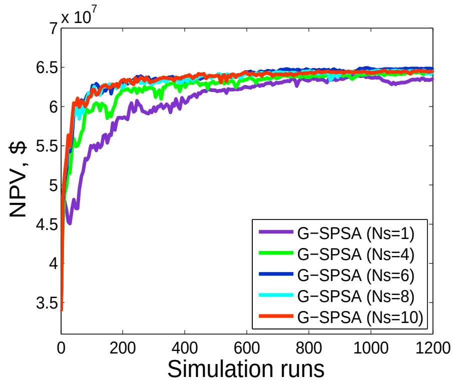


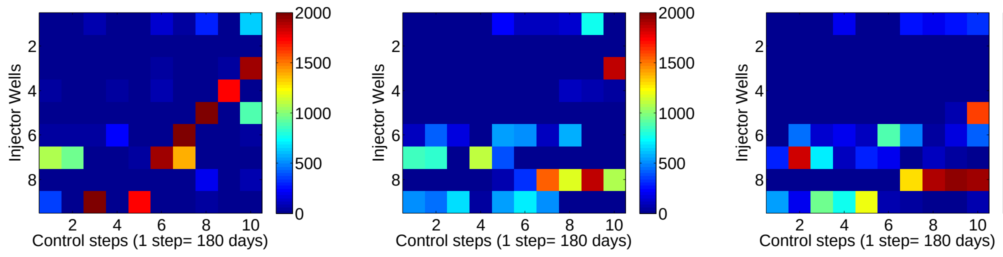
Figure 2.5: NPV versus the number of simulation runs with different correlation lengths, N_s.

Table 2.6: Average NPV obtained from five runs of each algorithm with five different initial random seeds when we change correlation lengths, N_s ($M = 5$, $k_{\max} = 200$, $A = 20$, $a_0 = 2.0$, $c_0 = 0.1$; and $\sigma^2 = (1, 1)$).

Algorithm	No. of simulations	Final NPV,$\times 10^7$\$
G-SPSA ($N_s = 1$)	1200	6.102
G-SPSA ($N_s = 4$)	1200	6.383
G-SPSA ($N_s = 6$)	1200	6.434
G-SPSA ($N_s = 8$)	1200	6.411
G-SPSA ($N_s = 10$)	1200	6.433

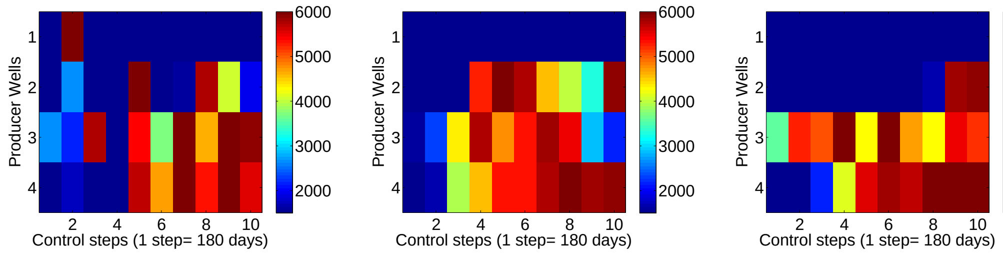
From the two preceding comparisons and other results not shown, we find that the G-SPSA algorithm's performance is virtually always better than that of the B-SPSA algorithm, so from this point we focus on the G-SPSA algorithm. We first investigate the effect of the values of variances and correlation length in the covariance matrix, C_U (Eq. 2.15), on the G-SPSA algorithm.

In considering the effect of correlation length, N_s, we use the average of five G-SPSA gradients calculated by one-sided simultaneous perturbations and apply the steepest-ascent direction of Eq. 2.21 to update the control variables. We emphasize



(a) $N_s = 1$ (b) $N_s = 6$ (c) $N_s = 10$

Figure 2.6: The estimated optimal water injection rate well controls obtained from G-SPSA when we change correlation lengths, N_s ($M = 5$, $k_{\max} = 200$, $A = 20$, $a_0 = 2.0$, $c_0 = 0.1$; and $\sigma^2 = (1, 1)$).



(a) $N_s = 1$ (b) $N_s = 6$ (c) $N_s = 10$

Figure 2.7: The estimated optimal BPH controls obtained from G-SPSA when we change correlation lengths, N_s ($M = 5$, $k_{\max} = 200$, $A = 20$, $a_0 = 2.0$, $c_0 = 0.1$; and $\sigma^2 = (1, 1)$).

again that optimization is always done in the log-transform domain. The control variables are perturbed with the Gaussian distribution which has a pair of variances (1,1), of which the first number corresponds to the transformed variable of BHP control and the second number corresponds to the transformed variable of water injection rate control. Other parameters are the same as used in the two previous comparisons, i.e, $M = 5$, $a_0 = 2.0$ and $c_0 = 0.1$. The relationship between the NPV versus the number of simulation runs is shown in Fig. 2.5 for the various correlation lengths. When the correlation length is equal to 1, there is no temporal (or spatial) correlation between control variables. Note that $N_s = 1$ yields the lowest NPV as well as the lowest convergence rate. In addition, $N_s = 1$ results in the roughest optimal well controls (see Figs. 2.6 and 2.7). The correlation lengths $N_s = 6$, $N_s = 8$, and

$N_s = 10$ result in similar NPV's as well as similar convergence rates, and perform better than $N_s = 4$. From Figs. 2.6 and 2.7, we can see that the longer correlation length is, the smoother optimal well controls are. The average NPV obtained from five different runs of each algorithm with five different initial random seeds is shown in Table 2.6 when we change correlation lengths. The correlation length of $N_s = 6$ results in the best NPV of \$$6.434 \times 10^7$ which is about 5.4% greater than that obtained from correlation length of $N_s = 1$. From the results of Table 2.6, we can see that there is not a great difference in the NPV's as long as the correlation length satisfies $N_s \geq 4$. The estimated optimal control vector becomes smoother in time as N_s increases (see Figs. 2.6 and 2.7). However, if we keep increasing the correlation length, we effectively decrease the variability in time which can prevent us from obtained an optimal solution if the optimal well controls is a bang-bang optimal solution, where, for example, the optimal water injection rate can be the maximum allowable rate at one control step and zero at the next control step.

As we work in the log-transform domain, it is less clear what a good value of variance should be. If $s_i = \ln\left(\frac{u_i - u_i^{\text{low}}}{u_i^{\text{up}} - u_i}\right)$, $i = 1, 2, ..., n_{\text{u}}$, close to zero, choosing perturbation on the order of 0.1 might be appropriate so that $\delta J = J(s + c_k \delta s) - J(s)$ is meaningful. But if s_i is equal to 20, it may require a variance on the order of 15 to ensure that the calculation of δJ is numerically meaningful. We might expect truncation might alleviate the problem in the latter cases. Note, however, that the stochastic gradient is

$$\widehat{g}_k = \frac{J(u_k + c_k \delta u_k) - J(u_k)}{c_k} \delta u_k, \tag{2.30}$$

where we have now returned to using u as the optimization variables after applying the log-transformation. In Eq. 2.30, δu_k is defined by

$$\delta u_k = C_U^{1/2} Z_k, \tag{2.31}$$

where $Z_k \sim N(0, I)$. Letting $\overline{\widehat{g}}_k$ be the average of several stochastic gradients, G-SPSA is given by

$$u_{k+1} = u_k + a_k \frac{\overline{\widehat{g}}_k}{\|\overline{\widehat{g}}_k\|_\infty}. \tag{2.32}$$

Eq. 2.32 indicates the magnitude of the change in any component of u is controlled by a_k. So the choice of variance is used to define C_U affect only the accuracy of

$$\delta J_k = J(u_k + c_k \delta u_k) - J(u_k). \tag{2.33}$$

As we increase the variance in defining C_U with c_k fixed, the difference between $E[\widehat{g}_k]$ and $\nabla J(u_k)$ increases but the computation of δJ_k is less likely to be inaccurate due to cancellation of leading digits when performing the subtraction on the right hand side of Eq. 2.33. Note what determines the accuracy of δJ_k is the magnitude of $c_k \delta u_k$, i.e.,

$$c_k \|\delta u_k\| = c_k \|C_U^{1/2} Z_k\|, \tag{2.34}$$

and of course the sensitivity of J to change in u_k. As any change in the variance in C_U can be compensated by changing c_k, we simply set all variances in C_U equal to 1 and investigate the effect of c_k on the results. As the values of the c_k's are determined by the values of c (see Eq. 2.20) and the value of c is determined by the value of $c_{\min}$ (see the discussions above Eq. 2.20), we only need to investigate the effect of $c_{\min}$ on the results.

Li and Reynolds (2011) show that the stochastic gradient of Eq. 2.4 gives an uphill direction for sufficiently small c_k. The goal of this section is to find how small $c_{\min}$ should be to result in a better NPV as well as a better convergence rate. We keep the maximum number of allowable iterations equal to 200, which means the value of A is equal to 20. The initial step size of a_0 is set equal to 2.0, so we obtain the value of $a = 13$. The transformed variables are perturbed using Gaussian distribution with a variance pair of (1,1), of which the first number corresponds to the transformed variable of BHP control and the second number corresponds to the transformed variable of water injection rate control, and a correlation length of $N_s = 10$. The steepest-ascent direction is calculated by an average of five G-SPSA gradients. The normalized search direction (Eq. 2.21) is applied to update the control vector. We choose a various values of $c_{\min}$ (Table. 2.7) to run experimental computations with the hope that we can find a common way for choosing a reasonable minimum allowable change in perturbation size. Fig. 2.8 shows the relationship between NPV versus the number of reservoir simulation runs (obtained from one optimization run of each algorithm with one initial random seed) when we use different values of $c_{\min}$. The results in Fig. 2.8 show that there is little difference between NPV's obtained from a variety of $c_{\min}$. Table. 2.7 presents the G-SPSA algorithm's performance using different values of $c_{\min}$. The average NPV's in Table. 2.7 are obtained from five runs of each algorithm with five different initial random seeds. The value of $c_{\min} = 0.1$ results in the best NPV of \$$6.482 \times 10^7$ which is about 2.7% greater than that resulting from $c_{\min} = 0.0001$ and is about 1.6% greater than that resulting from $c_{\min} = 1.0$. Note that the NPV is not extremely sensitive to the value of $c_{\min}$.

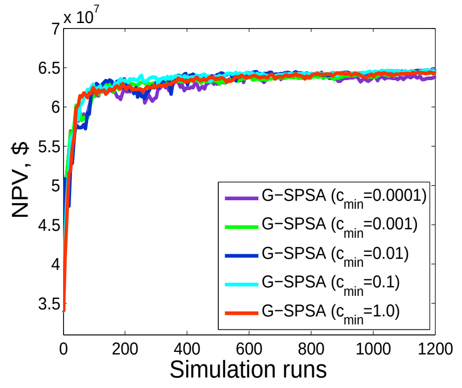


Figure 2.8: NPV versus the number of simulation runs with different values of $c_{\min}$.

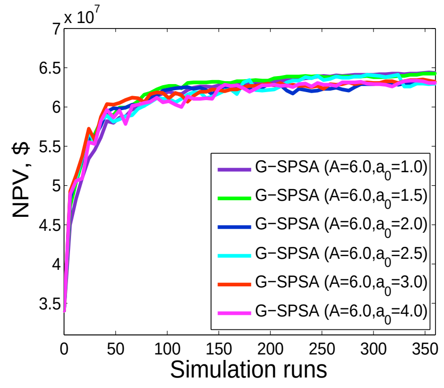


Figure 2.9: NPV versus the number of simulation runs with $A = 6$ and different values of a_0 .

We know that SPSA parameters in Eq. 2.19 effect the SPSA algorithm's

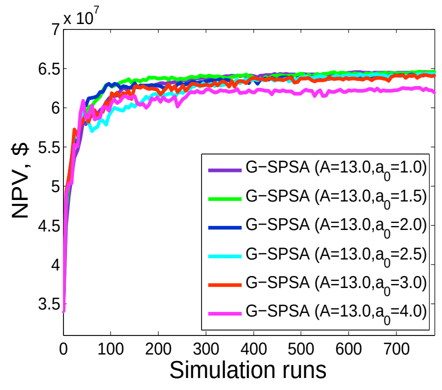


Figure 2.10: NPV versus the number of simulation runs with $A = 13$ and and different values of a_0.

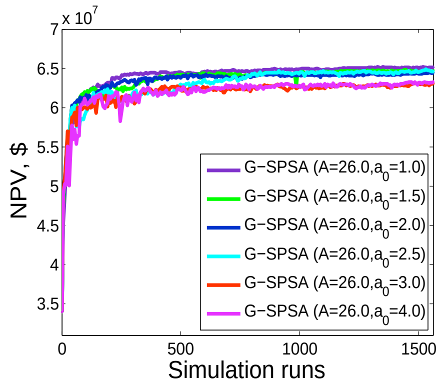


Figure 2.11: NPV versus the number of simulation runs with $A = 26$ and different values of a_0.

Table 2.7: Average NPV obtained from five runs of each algorithm with five different initial random seeds when we use different values of $c_{\min}$ ($M = 5$, $k_{\max} = 200$, $A = 20$, $a_0 = 2.0$; and $N_s = 10$ and $\sigma^2 = (1, 1)$) .

Algorithm	No. of simulations	c_0	Final NPV,$\times 10^7$\$
G-SPSA ($c_{\min} = 0.0001$)	1200	0.000178	6.310
G-SPSA ($c_{\min} = 0.001$)	1200	0.00178	6.424
G-SPSA ($c_{\min} = 0.01$)	1200	0.0178	6.436
G-SPSA ($c_{\min} = 0.1$)	1200	0.178	6.482
G-SPSA ($c_{\min} = 1.0$)	1200	1.78	6.378

performance. However, it is hard to investigate a general way to estimate a good value of A as well as a. Spall (2003) suggests that experimental computation is a suitable method to calculate these values in each specified case study. In the next part of this example, we will discuss the experimental computation to estimate some SPSA parameters such as A and a for a production optimization problem with only bound constraints that was converted into the unbounded control variables using the log-transformation. To do this, we use an average of five G-SPSA gradients to approximate the steepest ascent direction. The controls are sampled from a Gaussian distribution with a variance pair of (1,1), of which the first number corresponds to the transformed variable of BHP control and the second number corresponds to the transformed variable of water injection rate control, and a correlation length of $N_s = 10$. The normalized search direction is used to update the unbounded control variables (Eq. 2.21). As discussed previously, as the maximum number of allowable iterations changes, the value of $c_{\min}$ is adjusted but we keep constant (see discussion above Eq. 2.20). Based on the results of Table. 2.7, we fix $c_0 = 0.1$ regardless of the maximum number of allowable iterations specified.

Table 2.8: Average NPV obtained from five runs of each algorithm with five different initial random seeds when we use different values of A and a_0 ($M = 5$, and $c_0 = 0.1$; and $N_s = 10$ and $\sigma^2 = (1, 1)$).

Stepsize a_0	$A = 6.0$		$A = 13.0$		$A = 26.0$	
	Simul-	Final NPV	Simul-	Final NPV	Simul-	Final NPV
$a_0 = 1.0$	360	6.342×10^7	780	6.419×10^7	1560	6.442×10^7
$a_0 = 1.5$	360	6.351×10^7	780	6.423×10^7	1560	6.470×10^7
$a_0 = 2.0$	360	6.324×10^7	780	6.398×10^7	1560	6.449×10^7
$a_0 = 2.5$	360	6.354×10^7	780	6.399×10^7	1560	6.364×10^7
$a_0 = 3.0$	360	6.251×10^7	780	6.334×10^7	1560	6.335×10^7
$a_0 = 4.0$	360	6.212×10^7	780	6.254×10^7	1560	6.206×10^7

To choose a good value of A, we normally set A equal to 10% of the maximum number of allowable iterations. However until now, we do not have any information to choose the maximum number of allowable iterations for each specified problem because each production optimization problem may have a different optimal value. After working with a lot of experimental computations, we find the information for choosing this value might come from the total number of control variables. In this example, the total number of control variables is equal to 130. Therefore, we tentatively choose three values of the maximum allowable iterations:

- $k_{\max} = 60$, which is about one half the total number of control variables, and then we obtain A=6. The maximum allowable simulation runs is equal to $60 \times 6 = 360$, because the computation of each search direction is based on the average of five stochastic gradients requires six reservoir simulation runs as we use one-sided simultaneous perturbation.

- $k_{\max} = 130$, which is identical to the total number of control variables, and then we obtain A= 13. The maximum allowable simulation runs is equal to $130 \times 6 = 780$.

- $k_{\max} = 260$ which is double the total number of control variables, and then we obtain A= 26. The maximum allowable simulation runs is equal to $260 \times 6 = 1560$.

We consider various values of initial stepsize $a_0 = 1.0$, $a_0 = 1.5$, $a_0 = 2.0$ $a_0 = 2.5$, $a_0 = 3.0$, and $a_0 = 4.0$ to run the G-SPSA algorithm. Figs. 2.8, 2.9, and 2.10 respectively present the relationship of NPV and the number of simulation runs corresponding to A=6, A=13, and A=26. Note that, in these three figures (Figs. 2.8, 2.9, and 2.10), we plot the NPV obtained from one optimization with one initial random seed. We can observe the same trend in three different values of A is that the biggest value of a_0 in this example gives the worst value of NPV as well as convergence rate.

Table 2.8 presents the algorithm's performance with different values of A and a_0 at different maximum numbers of allowable reservoir simulation runs. In all cases, A is equal to 10% of the maximum number of reservoir simulation allowable. Note that Table. 2.7 presents the average NPV's obtained from five runs of each algorithm with five different initial random seeds. The values of "simul" in Table 2.8 gives the number of simulation runs corresponding to the maximum number of allowable iterations, i.e., in the $k_{\max} = 60$ case (simul= 360), the $k_{\max} = 130$ case (simul= 780) and the $k_{\max} = 260$ case (simul= 1560). The NPV values corresponding to those values of "simul" will be compared against other results. Overall, $a_0 = 1.5$ gives the best results but there is not much variation in the NPV's for the three cases

$a_0 = 1.5$, $a_0 = 2.0$, and $a_0 = 2.5$. Moreover, for each of these three values of a_0, the NPV increases as the maximum number of allowable iterations increases.

Table 2.9 gives the comparison of results based on stopping simulation runs at 1200 in all $k_{\max}$ cases, i.e., we actually let iterations continue beyond the value of $k_{\max}$ when $k_{\max} = 60$ and $k_{\max} = 130$. The NPV values corresponding to those obtained after 1200 reservoir simulation runs are shown in Table 2.9. We can see a suitable value of initial step szie a_0 should be in the range from 1.0 to 2.5.

Table 2.9: Average NPV obtained from five runs of each algorithm with five different initial random seeds at fixed 1200 reservoir simulation runs when we use different values of A and a_0 ($M = 5$, and $c_0 = 0.1$; and $N_s = 10$ and $\sigma^2 = (1, 1)$).

Stepsize a_0	Simulations	$A = 6.0$ (Final NPV)	$A = 13$ (Final NPV)	$A = 26$ (Final NPV)
$a_0 = 1.0$	1200	6.445×10^7	6.442×10^7	6.440×10^7
$a_0 = 1.5$	1200	6.486×10^7	6.464×10^7	6.464×10^7
$a_0 = 2.0$	1200	6.429×10^7	6.372×10^7	6.429×10^7
$a_0 = 2.5$	1200	6.457×10^7	6.429×10^7	6.364×10^7
$a_0 = 3.0$	1200	6.381×10^7	6.356×10^7	6.326×10^7
$a_0 = 4.0$	1200	6.300×10^7	6.339×10^7	6.217×10^7

Before going to conclusion, we set up two cases of $k_{\max}$ ($k_{\max} = 130$ and $k_{\max} = 200$) with other optimal parameters coming from the previous analysis to see how these parameters perform the G-SPSA algorithm (see Table 2.10). The results in Table 2.10 show that with optimal parameters based on the previous results, the G-SPSA algorithm corresponding to each $k_{\max}$ always gives the best NPV.

Finally, we can draw some conclusions from this example. The results from Table 2.2 to Table 2.10 are based on the average value of NPV from five optimizations corresponding to five different initial random seeds. Therefore, the following conclusions are be reasonable.

- The one-sided simultaneous perturbation with both Bernoulli distribution and

Table 2.10: Average NPV obtained from five runs of each algorithm with five different initial seeds When we use two values of $k_{\max}$ and other optimal paramtes.

Optimal parameters	No. of simulations	Final NPV,$\times 10^7$\$
$M = 5$, $k_{\max} = 130$, $A = 13$, $a_0 = 1.5$ and $c_{\min} = 0.1$; $N_s = 6$, $\sigma^2 = (1, 1)$	780	6.442
$M = 5$, $k_{\max} = 200$, $A = 20$, $a_0 = 1.5$ and $c_{\min} = 0.1$; $N_s = 6$, $\sigma^2 = (1, 1)$	1200	6.487

Gaussian distribution results in a more computationally efficient algorithm than is obtained with a two-sided simultaneous perturbation;

- The performance of the algorithm with Gaussian perturbation is always better than the algorithm's performance with Bernoulli perturbation;

- The performance of ComG-SPSA1 and ComG-SPSA2 is not better than the G-SPSA algorithm's performance;

- For both Bernoulli perturbation and Gaussian perturbation, optimal number (M) of SPSA gradients to use in construction the search direction on steepest-ascent algorithm is on the order of 5 to 10;

- When we use the Gaussian perturbation, a small correlation length results in very rough final well controls (which may not be very practical). Larger correlation lengths give smoother final control values. Based on our experimentations, the correlation length should be at least one half the number of the control steps. However, the correlation length equal to the number of control step is not suggest as it leads to a very smooth and sub-optimal final control variables.

- We can say that a good value of $c_{\min}$ is from 0.01 to 1.0 for this example;

- In a production optimization problem which has an undefined given number of control variables, the maximum allowable iteration should be greater or equal to the total number of control variables, say 1 to 2 times the number of control variables if sufficient computation time is available. We can set the initial stepsize in the range from 1.0 to 2.0.

2.5.2 Optimization with simple bound constraints for PUNQ case

Here, we first consider the PUNQ-S3 reservoir (Floris et al., 2001; Barker et al., 2001; Gao et al., 2006; Zhao et al., 2011). PUNQ-S3 is a three-phase, three-dimensional reservoir simulation model. The well locations for producers and injectors are shown in the layer-1 horizontal log-permeability field in Fig. 2.12. The reservoir has five simulation layers and it is bounded to the east and north by faults. A small gas cap is located in the center of the dome shaped structure. The original model has a fairly strong aquifer at the south and west sides, but in the model considered here, the aquifer has been eliminated and water injection wells have been added as shown in Fig. 2.12. There are 7 vertical producing wells and 7 vertical water injection wells. The reservoir life is set to 7600 days. The length of each control step is set to 190 days. Thus, there are 40 control steps and the total number of control variables is equal to $(7 + 7) \times 40 = 560$. During optimization, the injection wells are placed under water injection rate control with a lower bound of 0 STB/D and an upper bound of 5000 STB/D. The producer wells are placed under bottom hole pressure (BHP) control with a lower bound of 1000 psi and an upper bound of 3000 psi. The oil price is \$80.0/STB, the water production cost is \$8.9/STB, the water injection cost is \$0.0/STB, and the annual discount rate is 0.

The initial guess for the well controls is equal to 2500 STB/D for each injection rate control and 1500 psi for each producer BHP control. In this example, the log-transformation is still applied to convert the bound constraints into the un-

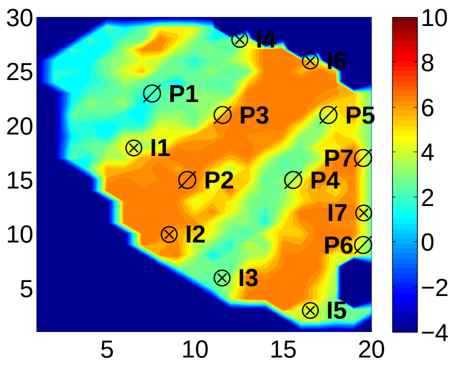


Figure 2.12: Well locations (Layer-1 horizontal log permeability field).

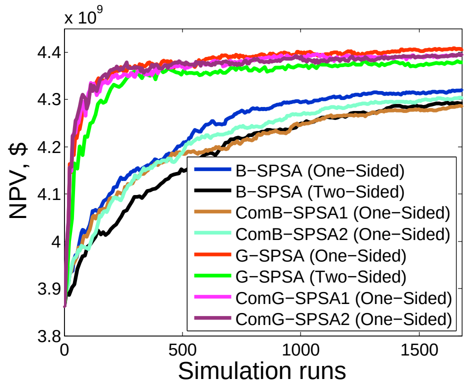


Figure 2.13: NPV versus the number of simulation runs.

bounded control variables. During optimization, we work in the log-transform domain. However, the transformed control variables will be converted back the true control variables to use in simulator. The algorithm terminates when the number of iterations reaches the maximum allowable value.

In this example, the performance of the SPSA algorithms using the Bernoulli distribution will be compared with the algorithm's performance using the Gaussian distribution. To do this, we apply an average of five SPSA gradients ($M = 5$) estimated by one-sided simultaneous perturbation or two-sided simultaneous perturbation (Eqs. 2.2, 2.3, 2.4 and 2.5) to approximate the steepest ascent direction. The normalized search direction (Eq. 2.21) is used to update control variables for all SPSA algorithms. For the G-SPSA algorithms, control variables are sampled from Gaussian distribution based on a Spherical covariance function with a variance pair of (1,1), of which the first number corresponds to the transformed variable of BHP control and the second number corresponds to the transformed variable of water injection rate control, and a correlation length of $N_s = 40$ (Eq. 2.15). Note that this is a very long correlation length and is expected to yield smooth well controls. The value of α and γ in these equations is similar to the value suggested by Spall (1998), $\alpha = 0.602$ and $\gamma = 0.101$. By setting the maximum number of allowable iterations equal to 280, which is one half the total number of control variables (560), we obtain the value of $A = 28$. When the value of M is equal to 5, we need six reservoir simulation runs to obtain an approximate gradient and an updated NPV value. So the maximum number of allowable simulation runs with one-sided simultaneous perturbation is equal to $6 \times 280 = 1680$. By setting the value of a_0 equal to 2.0, we obtain the value of $a = 15.5$. We set $c_{\min}$ equal to 0.05658 which gives $c = 0.1$.

Table 2.11: Average NPV obtained from five runs of each algorithm with five different initial random seeds ($M = 5$, $k_{\max} = 280$, $A = 28$, $a_0 = 2.0$, $c_0 = 0.1$; $N_s = 40$ and $\sigma^2 = (1, 1)$).

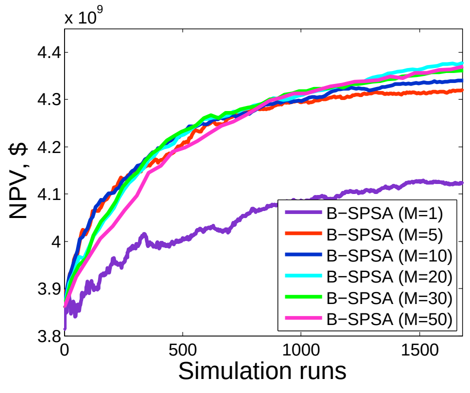
Algorithm	No. of simulations	Final NPV,$\times 10^9$\$
B-SPSA(One-Sided)	1680	4.298
B-SPSA(Two-Sided)	1680	4.297
ComB-SPSA1(One-Sided)	1680	4.317
ComB-SPSA2(One-Sided)	1680	4.308
G-SPSA(One-Sided)	1680	4.398
G-SPSA(Two-Sided)	1680	4.386
ComG-SPSA1(One-Sided)	1680	4.400
ComG-SPSA2(One-Sided)	1680	4.392

Table 2.12: Average NPV obtained from five runs of each algorithm with five different initial seeds when we change values of $k_{\max}$ at fixed number of simulation runs, 1680 ($M = 5$, $a_0 = 2.0$, and $c_0 = 0.1$; $N_s = 40$ and $\sigma^2 = (1, 1)$).

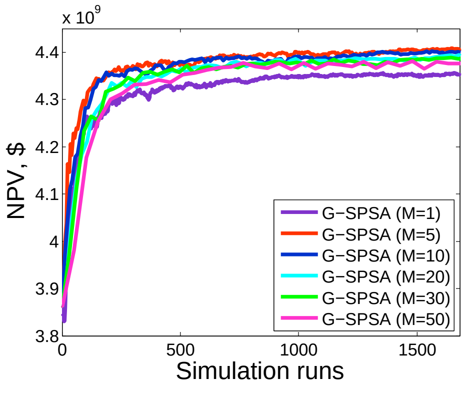
Algorithm	Simulations	$k_{\max}$	A	Final NPV,$\times 10^9$\$
B-SPSA(One-Sided)	1680	$k_{\max} = 280$	$A = 28$	4.298
B-SPSA(Two-Sided)	1680	$k_{\max} = 152$	$A = 15$	3.987
G-SPSA(One-Sided)	1680	$k_{\max} = 280$	$A = 28$	4.398
G-SPSA(Two-Sided)	1200	$k_{\max} = 152$	$A = 15$	4.375

Fig. 2.13 presents NPV versus the number of simulation runs resulting from one optimization run with one initial random seed. Note that when the average of five SPSA gradients is calculated by two-sided simultaneous perturbation using Eqs. 2.3

and 2.5, corresponding to the maximum number of allowable iterations equal to 280, we obtain the total number of reservoir simulation runs is equal to $11 \times 280 = 3080$. However, we still compare the NPV values after 1680 simulation runs. The G-SPSA algorithm using one-sided simultaneous perturbation gives the highest final NPV and convergence rate. Similar to example 1, the NPV's obtained from the SPSA algorithm using one-sided simultaneous perturbation is always greater than those obtained from the SPSA algorithm using two-sided simultaneous perturbation. The results of Fig. 2.13 suggest that the NPV of the G-SPSA algorithm increases faster than that of the B-SPSA algorithm. Table 2.11 presents the SPSA algorithm's performance when we calculate the average NPV's from five runs of each algorithm with five different initial random seeds. In this example, the one-sided simultaneous perturbation results in a better NPV than the two-sided simultaneous perturbation. As in example 1, the performance of each SPSA algorithm with Gaussian perturbation is always better than the performance of the corresponding SPSA algorithm with Bernoulli perturbation. The ComG-SPSA1 algorithm using one-sided simultaneous perturbation gives the best final NPV of $\$4.400 \times 10^9$ but this is essentially the same as the NPV of $\$3.398 \times 10^9$ obtained with one-sided simultaneous perturbation. Again, to make the above conclusion more strong, we run the SPSA algorithm using two-sided simultaneous perturbation with another maximum number of allowable iterations, $k_{\max} = 152$, which corresponds to the fixed number of simulation runs equal to 1680 and the number of SPSA gradients equal to 5 ($M = 5$). Because when we use the value of $M = 5$, at each iteration, the SPSA algorithm using two-sided simultaneous perturbation needs ($2 \times M + 1 = 11$) reservoir simulation runs to generate a steepest-ascent direction, and to obtain an updated NPV value. With $k_{\max} = 152$, we obtain the value of $A = 15$. Note that the initial stepsize and perturbationsize is kept constant, i.e., $a_0 = 2.0$ and $c_0 = 0.1$. The average NPV's obtained from five runs of each algorithm with five different initial random seeds in Table 2.12 show



(a) B-SPSA



(b) G-SPSA

Figure 2.14: NPV versus the number of simulation runs obtained from one optimization run with one initial random seed when we change values of M ($k_{\max} = 280$, $A = 28$, $a_0 = 2.0$, $c_0 = 0.1$; $N_s = 40$ and $\sigma^2 = (1, 1)$).

that the SPSA algorithm using one-sided simultaneous perturbations works better than that using two-sided simultaneous perturbation. This is consistent with example 1, so from this point on we focus on the algorithms using one-sided simultaneous perturbation.

Another comparison is done using different numbers of SPSA gradients, M. Except for changing the number of SPSA gradients which are averaged to compute the average stochastic gradient $\overline{\widehat{g}}_k$ (see Eq. 2.17), all other parameters are kept the same, i.e., $A = 280$, $a_0 = 2.0$ and $c = 0.1$. The normalized search direction (Eq. 2.21) is used to update control variables. When the number of SPSA gradient changes, the total number of simulation runs required to compute $\overline{\widehat{g}}_k$ changes, however, the NPV obtained from each M value at the 1680^{th} simulation run is still used to compare results. Fig. 2.14 shows the relationship between NPV and the number of simulation runs. The average NPV 's obtained from five runs of each algorithm with five different initial seeds are presented in Table 2.13. From Fig. 2.14 and Table 2.13, we see that the relationship between NPV and the number of simulation runs in this example is similar to that of example 1. For the B-SPSA algorithm, when the value of M is equal to 1, we obtain the worst NPV of $\$4.138 \times 10^9$. When the value of M is equal to 20, we obtain the largest NPV of $\$4.370 \times 10^9$, however, this value is smaller than the NPV's obtained with the G-SPSA algorithm with $M = 5$ and $M = 10$. Thus as in example 1, G-SPSA with $M = 5$ and $M = 10$ gives a better NPV than B-SPSA with any value of M. To make the above conclusion more logical, we again fix the total number of reservoir simulation runs at 1680, corresponding to each value of M, we obtain one value of $k_{\max}$ which is used to estimate the value of A in Eq. 2.19. The SPSA algorithm using one-sided simultaneous perturbations is applied to generate an approximate gradient, therefore, at each iteration we need $M+1$ reservoir simulation runs to generate a steepest-ascent direction, and to obtain an updated NPV value. Corresponding to each value of M, we obtain a maximum number of allowable iterations by roughly using the following equation: $k_{\max} = 1680/(M+1)$ (see Table 2.14). We again keep the initial step size and perturbation size constant, i.e., $a_0 = 2.0$ and $c_0 = 0.1$. Fig. 2.15 presents NPV versus the number of simulation runs of the G-SPSA algorithm as the value of M used to generate the average stochastic gradient $\overline{\widehat{g}}_k$ (see Eq. 2.17) changes as well as the corresponding $k_{\max}$ also changes. The average NPV 's obtained from five runs of each algorithm with five different initial seeds are presented in Table 2.14. The results from Fig. 2.15 and Table 2.14 show that the value of $M = 5$ always gives better NPV results at 1680 reservoir simulation runs than other values of M when we use the corresponding maximum number of allowable iterations.

The next step in this example is to find a reasonable correlation length, N_s. Therefore, we consider various values of correlation lengths to run the G-SPSA algorithm (see Table 2.15). We keep other SPSA parameters constant, i.e., $M = 5$, $k_{\max} = 280$, $a_0 = 2.0$, and $c_0 = 0.1$. Control variables are sampled from Gaussian distribution based on a Spherical covariance function with a variance pair of (1,1), of which the first number corresponds to the transformed variable of BHP control and the second number corresponds to the transformed variable of water injection rate

Table 2.13: Average NPV obtained from five runs of each algorithm with five different initial random seeds when we change the value of M ($k_{\max} = 280$, $A = 28$, $a_0 = 2.0$, $c_0 = 0.1$; $N_s = 40$ and $\sigma^2 = (1, 1)$).

Algorithm	No. of simulations	Final NPV,$\times 10^9$\$
BSPSA ($M = 1$)	1680	4.138
BSPSA ($M = 5$)	1680	4.298
BSPSA ($M = 10$)	1680	4.340
BSPSA ($M = 20$)	1680	4.370
BSPSA ($M = 30$)	1680	4.362
BSPSA ($M = 50$)	1680	4.369
GSPSA ($M = 1$)	1680	4.378
GSPSA ($M = 5$)	1680	4.398
GSPSA ($M = 10$)	1680	4.397
GSPSA ($M = 20$)	1680	4.399
GSPSA ($M = 30$)	1680	4.394
GSPSA ($M = 50$)	1680	4.389

control. The G-SPSA algorithm using one-sided simultaneous perturbation is used to generate the steepest-ascent direction. The normalized search direction (Eq. 2.21) is used to update control variables. The NPV values versus the number of reservoir simulation runs when the value of N_s changes is presented in Fig. 2.16. In Fig. 2.16, there are five curves corresponding to five different values of N_s, these curves shows that NPV is a function of simulation runs. When there are no correlation among well controls at different control steps, i.e., N_s is equal to 1, the NPV obtained is the worst value. When we increase the correlation length of N_s, the NPV and convergence rate tends to increase. However, the full correlation length does not give the best NPV as well as the convergence rate. Figs. 2.17 and 2.18 present the final well controls obtained from the G-SPSA algorithm using different values of N_s. We see that $N_s = 1$, i.e., there is no temporal (or spatial) correlation between control variables, yields the roughest final well controls. Similar to example 1, as we increase the value of N_s, the final well controls become more smooth. The final well controls is the most smooth when the full correlation length, $N_s = 40$, is applied (see

Table 2.14: Average NPV obtained from five runs of each algorithm with five different initial random seeds when we change the value of M and the corresponding value of $k_{\max}$ at fixed number of simulation runs, 1680 ($a_0 = 2.0$, $c_0 = 0.1$; $N_s = 40$ and $\sigma^2 = (1, 1)$).

Algorithm	Simulations	$k_{\max}$	A	Final NPV,$\times 10^9$\$
B-SPSA ($M = 1$)	1680	$k_{\max} = 840$	$A = 84$	3.882
B-SPSA ($M = 5$)	1680	$k_{\max} = 280$	$A = 28$	4.298
B-SPSA ($M = 10$)	1680	$k_{\max} = 152$	$A = 15$	3.909
B-SPSA ($M = 20$)	1680	$k_{\max} = 80$	$A = 8$	4.056
B-SPSA ($M = 30$)	1680	$k_{\max} = 54$	$A = 5$	4.100
B-SPSA ($M = 50$)	1680	$k_{\max} = 32$	$A = 3$	4.082
G-SPSA ($M = 1$)	1680	$k_{\max} = 840$	$A = 84$	4.333
G-SPSA ($M = 5$)	1680	$k_{\max} = 280$	$A = 28$	4.398
G-SPSA ($M = 10$)	1680	$k_{\max} = 152$	$A = 15$	4.381
G-SPSA ($M = 20$)	1680	$k_{\max} = 80$	$A = 8$	4.383
G-SPSA ($M = 30$)	1680	$k_{\max} = 54$	$A = 5$	4.380
G-SPSA ($M = 50$)	1680	$k_{\max} = 32$	$A = 3$	4.371

Figs. 2.17 and 2.18). Repeating the experimental computation with five different initial random seeds, and then computing the average NPV gives the results shown in Table 2.15. In Table 2.15, we compare the NPV resulting from different values of N_s after 1680 simulation runs. The correlation length of $N_s = 20$ results in the best NPV of \$$4.418 \times 10^9$, which is slightly greater than the NPV resulting from $N_s = 40$.

Table 2.15: Average NPV obtained from five runs of each algorithm when we use different correlation lengths, N_s ($M = 5$, $k_{\max} = 280$, $A = 28$, $a_0 = 2.0$, $c_0 = 0.1$; and $\sigma^2 = (1, 1)$).

Algorithm	No. of simulations	Final NPV,$\times 10^9$\$
G-SPSA ($N_s = 1$)	1680	4.304
G-SPSA ($N_s = 10$)	1680	4.405
G-SPSA ($N_s = 20$)	1680	4.418
G-SPSA ($N_s = 30$)	1680	4.406
G-SPSA ($N_s = 40$)	1680	4.398

In example 1, we found that the value of $c_{\min}$ (Eq. 2.20) has some effect on

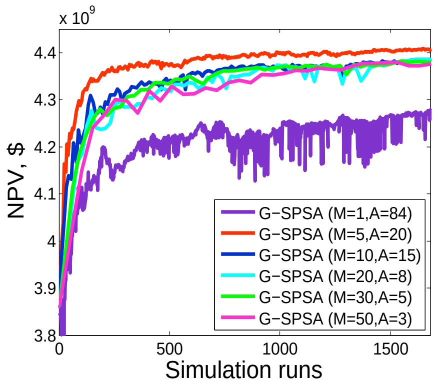


Figure 2.15: NPV versus the number of simulation runs with different values of M and $k_{\max}$.

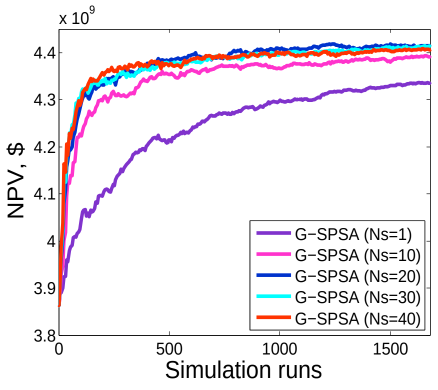
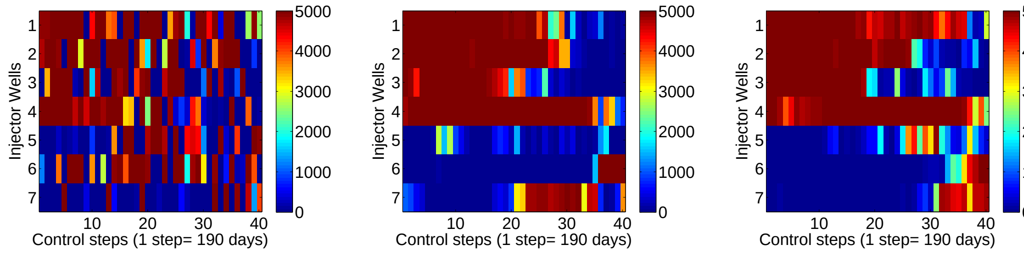


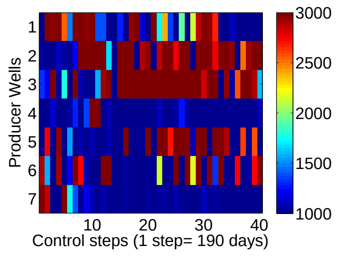
Figure 2.16: NPV versus the number of simulation runs with different correlation lengths, N_s.



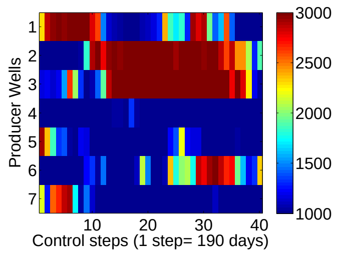
(a) $N_s = 1$ (b) $N_s = 20$ (c) $N_s = 40$

Figure 2.17: The estimated optimal water injection rate well controls obtained from G-SPSA when we change correlation lengths, N_s ($M = 5$, $k_{\max} = 280$, $A = 28$, $a_0 = 2.0$, $c_0 = 0.1$; and $\sigma^2 = (1, 1)$).

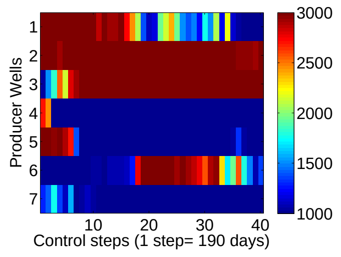
the performance of the G-SPSA algorithm. To find out a good value of $c_{\min}$ which is robust for any production optimization problems, we also use several different values of $c_{\min}$ to run the G-SPSA algorithm. After going through these results, a reasonable and robust $c_{\min}$ is found. As we know that the G-SPSA algorithm is working in log-transform domain, we tentatively choose various values of $c_{\min}$ from 0.0001 to 1.0. The G-SPSA algorithm using one-sided simultaneous perturbation is applied to generate an approximate gradient. We use a variance pair of (1,1), of which the first number corresponds to the transformed variable of BHP control and the second number corresponds to the transformed variable of water injection rate



(a) $N_s = 1$



(b) $N_s = 20$



(c) $N_s = 40$

Figure 2.18: The estimated optimal BPH controls obtained from G-SPSA when we change correlation lengths, N_s ($M = 5$, $k_{\max} = 280$, $A = 28$, $a_0 = 2.0$, $c_0 = 0.1$; and $\sigma^2 = (1, 1)$).

control, and a correlation length of $N_s = 40$ to generate the covariance matrix C_U (Eq. 2.15). Five G-SPSA gradients is used to average a steepest-ascent direction to update the control variables. The normalized search direction (Eq. 2.21) is used to update control variables. We choose the maximum number of allowable iterations equal to 280, i.e, the number of reservoir simulation runs is 1680, so A is equal to 28. We set the initial stepsize of a_0 equal to 2.0.

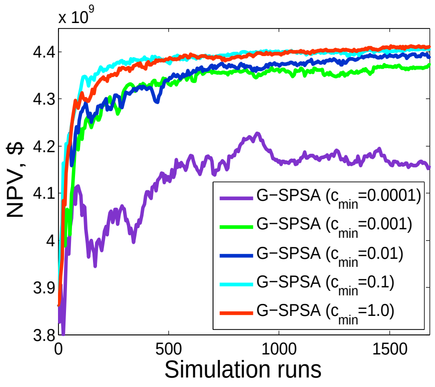


Figure 2.19: NPV versus the number of simulation runs with different values of $c_{\min}$.

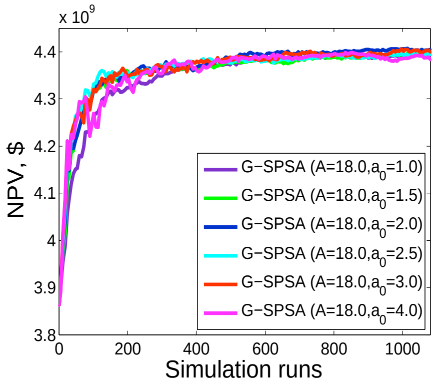


Figure 2.20: NPV versus the number of simulation runs with $A = 18$ and different values of a_0.

Fig 2.19 shows NPV versus the number of reservoir simulation runs when the value of $c_{\min}$ is adjusted. The value of $c_{\min} = 0.0001$ results in the worst NPV as well as the worst convergence rate. Otherwise, $c_{\min} = 1.0$ and $c_{\min} = 0.1$ give similar NPV's and convergence rate. Table 2.16 present the average NPV's obtained from five optimization runs with five different initial random seeds. There is a big difference among the NPV's whenever we change $c_{\min}$. In this example, we also realize that a small $c_{\min}$ cannot result in a better NPV as well as a better convergence rate. Significantly, $c_{\min} = 0.1$ results in the best NPV of \$4.422 $\times 10^9$, but any value of $c_{\min}$ between 0.01 and 1.0 gives similar results.

Table 2.16: Average NPV obtained from five runs of each algorithm when we use different values of $c_{\min}$ ($M = 5$, $k_{\max} = 280$, $A = 28$, $a_0 = 2.0$; and $N_s = 40$ and $\sigma^2 = (1, 1)$) .

Algorithm	No. of simulations	c_0	Final NPV,$\times 10^9$\$
G-SPSA ($c_{min} = 0.0001$)	1680	0.0001766	4.225
G-SPSA ($c_{min} = 0.001$)	1680	0.001766	4.372
G-SPSA ($c_{min} = 0.01$)	1680	0.01766	4.390
G-SPSA ($c_{min} = 0.1$)	1680	0.1766	4.409
G-SPSA ($c_{min} = 1.0$)	1680	1.766	4.411

To find an optimal initial stepzise a_0 (Eq. 2.19), we use the same procedure as in example 1. Howewver, in this example, the total number of control variable is equal to 560. We choose three different maximum numbers of allowable iterations, $k_{\max}$, and in each case set $A = 0.1k_{\max}$:

- $k_{\max} = 180$, which is around a third of the total number of control variables, which gives A=18. The maximum number of allowable simulation runs is equal to $180 \times 6 = 1080$, because throughout this example we average five G-SPSA gradients to obtain the search direction;

- $k_{\max} = 280$, which is equal to one half the total number of control variables, so that A= 28. The maximum number of allowable simulation runs is equal to $280 \times 6 = 1680$.

- $k_{\max} = 560$, which is identical to the total number of control variables, and then we obtain A= 26. The maximum number of allowable simulation runs is equal to $560 \times 6 = 3360$.

We consider six different initial stepsizes (Table 2.17) to run the G-SPSA algorithm. We again use the same parameters as we used in previous sections, i.e., $M = 5$, $\sigma^2 = (1, 1)$, and $N_s = 40$. Based on the results of Table. 2.16, we fix $c_0 = 0.1$ regardless of the maximum number of allowable iterations specified. The G-SPSA algorithm using one-sided simultaneous perturbation is applied to generate a steepest-ascent direction. Figs. 2.20, 2.21, and 2.22 (obtained from one optimization run with one initial random seed) respectively present the relationship of NPV and the number of simulation runs corresponding to A=18, A=28, and A=56. For each value of A, we see that the NPV's obtained from different initial step sizes are almost the same. Table. 2.17 presents the average NPV's obtained from five runs of each algorithm with five different initial random seeds. Overall, $a_0 = 1.5$ gives the best results but there is not much variation in the NPV's for the three cases $a_0 = 1.5$, $a_0 = 2.0$, and $a_0 = 2.5$. However, for each of these three values of a_0, the NPV increases little as the maximum number of allowable iterations increases.

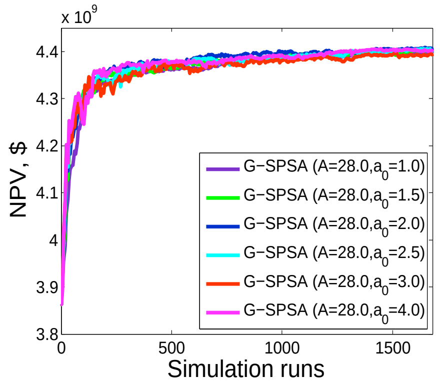


Figure 2.21: NPV versus the number of simulation runs with $A = 28$ and different values of a_0.

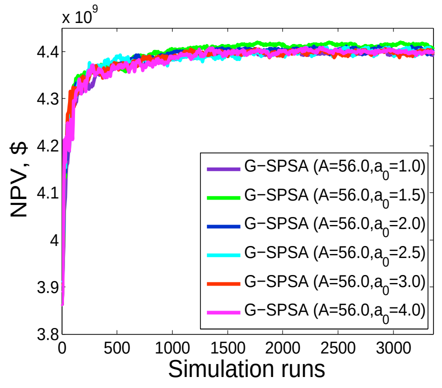


Figure 2.22: NPV versus the number of simulation runs with $A = 56$ and different values of a_0.

Table 2.18 gives the comparison of results based on stopping simulation runs at 1680 in all $k_{\max}$ cases, i.e., we actually let iterations continue beyond the value of $k_{\max}$ when $k_{\max} = 180$. The NPV values corresponding to each value of all $k_{\max}$ cases after 1680 reservoir simulation runs are shown in Table 2.8. Note that Table 2.18 presents the average NPV's obtained from five optimizations with five different initial random seeds. The results show that at least on this example the results are essentially independent of A and a_0 for the range of these variables that are considered. Perhaps the most important point is that we can.

Table 2.17: Average NPV obtained from five runs of each algorithm when we use different values of A and a_0 ($M = 5$ and $c_0 = 0.1$; and $N_s = 40$ and $\sigma^2 = (1, 1)$).

Stepsize a_0	$A = 18$		$A = 28$		$A = 56.0$	
	Simul-	Final NPV	Simul-	Final NPV	Simul-	Final NPV
$a_0 = 1.0$	1080	4.395×10^9	1680	4.402×10^9	3360	4.412×10^9
$a_0 = 1.5$	1080	4.395×10^9	1680	4.404×10^9	3360	4.410×10^9
$a_0 = 2.0$	1080	4.391×10^9	1680	4.398×10^9	3360	4.412×10^9
$a_0 = 2.5$	1080	4.391×10^9	1680	4.400×10^9	3360	4.403×10^9
$a_0 = 3.0$	1080	4.382×10^9	1680	4.401×10^9	3360	4.395×10^9
$a_0 = 4.0$	1080	4.387×10^9	1680	4.391×10^9	3360	4.401×10^9

Table 2.18: Average NPV obtained from five runs of each algorithm after 1680 reservoir simulation runs when we use different values of A and a_0 ($M = 5$ and $c_0 = 0.1$; and $N_s = 10$ and $\sigma^2 = (1, 1)$).

Stepsize a_0	Simulations	$A = 18.0$ (Final NPV)	$A = 28$ (Final NPV)	$A = 56$ (Final NPV)
$a_0 = 1.0$	1680	4.403×10^9	4.402×10^9	4.397×10^9
$a_0 = 1.5$	1680	4.402×10^9	4.404×10^9	4.401×10^9
$a_0 = 2.0$	1680	4.398×10^9	4.398×10^9	4.396×10^9
$a_0 = 2.5$	1680	4.398×10^9	4.400×10^9	4.389×10^9
$a_0 = 3.0$	1680	4.387×10^9	4.401×10^9	4.406×10^9
$a_0 = 4.0$	1680	4.401×10^9	4.391×10^9	4.394×10^9

Similar to example 1, we set up two cases of $k_{\max}$ ($k_{\max} = 280$ and $k_{\max} = 560$) with other optimal parameters coming from the previous analyzing to see how these parameters perform the G-SPSA algorithm (see Table 2.19). The results in Table 2.19 show that with optimal parameters based on the previous results, the G-SPSA algorithm corresponding to each $k_{\max}$ always gives the best NPV.

Table 2.19: Average NPV obtained from five runs of each algorithm with five different initial seeds When we use two values of $k_{\max}$ and other optimal paramtes.

Optimal parameters	No. of simulations	Final NPV,$\times 10^9$\$
$M = 5$, $k_{\max} = 280$, $A = 28$, $a_0 = 1.5$, and $c_{\min} = 0.5$; $N_s = 20$, $\sigma^2 = (1, 1)$	1680	4.421
$M = 5$, $k_{\max} = 560$, $A = 56$, $a_0 = 1.5$, and $c_{\min} = 0.5$; $N_s = 20$, $\sigma^2 = (1, 1)$	3360	4.444

Finally, we can draw some conclusions from experiments with this example. Generally what happened with the NPV and convergence rate in this example when SPSA parameters change is almost the same as in example 1. Based on the average NPV and the number of simulation runs obtained from five optimizing operations corresponding to five different initial random seeds, we can draw the following conclusions.

- The one-sided perturbation for both Bernoulli distribution and Gaussian distribution can give better results than the two-sided perturbation, this is the same conclusion obtained for example 1;

- The performance of algorithm with Gaussian distribution is always better than that with Bernoulli distribution, this is consistent with example 1;

- The performance of ComG-SPSA1 and ComG-SPSA2 is not any better than the G-SPSA algorithm's performance, this is also consistent with example 1.

- For Gaussian distribution, setting the number of SPSA gradients to generate the average stochastic search direction in range from 5 to 10 is optimal. For

Bernoulli distribution, $M = 5$ and $M = 10$ gave lower NPV's than the NPV from $M = 20$;

- When we use the Gaussian distribution, a small correlation length results in very rough final well controls (which may not be very practical). Larger correlation lengths give smoother final control values. Based on our experimentations, a correlation length on the order of one half the number of the control steps should be reasonable for this example;
- We can say that a good value of $c_{\min}$ is from 0.01 to 1.0 for this example, this is also consistent with example 1;
- For this example, setting the maximum number of allowable iterations equal to one half the number of control variables works as well as setting the maximum number of allowable iterations equal to the number of control variables. As in example 1, the initial stepsize from 1.0 to 2.5 works fine.

2.5.3 Optimization with simple bound constraints, Brugge case

The Brugge field is a synthetic reservoir developed by TNO (Peters et al., 2010) as a benchmark study to test different algorithms in the closed-loop reservoir management. The original model was constructed with approximately 20 million gridblocks and then upscaled to a 450,000 gridblock model, which is used as the true reservoir to provide observation data for history matching. The true case was used to construct data such as well logs and facies maps. According to this information, 104 geological realizations were upscaled to a 60,048 gridcell reservoir simulation model and provided to participants. The simulation model consists of nine layers, each with 139×48 gridblocks. The total number of active gridblocks is 44,550. The top structure map and well locations are in Fig. 2.23. Full details about the geological parameters of Brugge field are available in Peters et al. (2010).

The reservoir is bounded by faults and connected to a large aquifer. The reservoir has four geological layers and is further divided into 9 simulation layers. The Schelde formation corresponds to the top two simulation layers, the Waal formation corresponds to layer 3 through 5, the Maas formation corresponds to layer 6 to layer 8 and the Schie formation corresponds to the bottom layer. The structure of the field and fault descriptions were the same in all realizations. Each realization consisted of the following properties in each gridblock: net-to-gross (NTG), porosity, initial water saturation and absolute permeability in the x, y and z directions. Different geostatistical modeling methods were used for different realizations. For example, some realizations of properties in the first two layers were created with object-based modeling (channel objects in a shale background) whereas other realizations were created with sequential indicator simulation.

Throughout the reservoir life, only oil and water flow in the reservoir. Relative permeability and capillary pressure curves were provided by TNO (Peters et al., 2010) but with seven possible values of irreducible water saturation depending on the porosity of gridblock.

Here, we consider only the production optimization step of the closed-loop reservoir management problem. There are 30 vertical wells in the Brugge reservoir including 20 smart production wells and 10 smart water injection wells. Each well has multiple segments that can be controlled individually.

We optimize the well controls for years 10 through 30 based on the mean model obtained by Chen et al. (2010) using the ensemble Kalman filter with covariance localization to assimilate production data for the first ten years of the reservoir life (Fig. 2.24 and 2.25). This example is identical to the production optimization results for year 10-30 with an adjoint-gradient method which is reported in Chen et al. (2010).

The production period of the reservoir is divided into 40 control steps, i.e.,

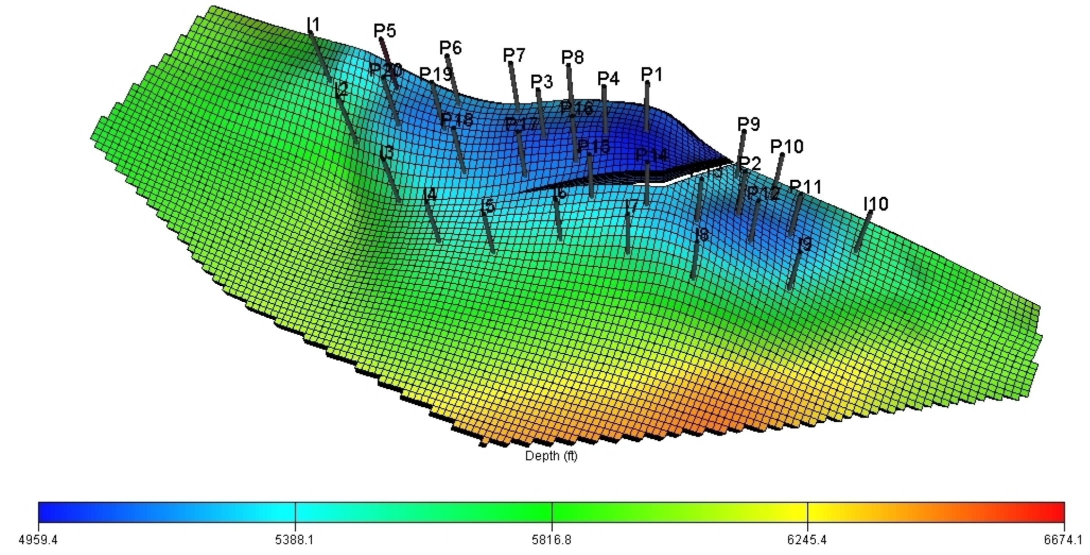


Figure 2.23: The top structure of Brugge field, Example 4.

each control step is 182.5 day. There are 84 control variables for each control step. These control variables are the liquid production rate at each individual segment of the production wells and the water injection rate at each individual segment of the injection wells. The total number of control variables is $40 \times 84 = 3360$. The maximum liquid production rate of each production segment is 3000 STB/D and, the maximum injection rate of each injector segment is 4000 STB/D. The minimum value for the rate of each segment in a production or an injection well is 0 STB/D. The minimum BHP constraint for a production segment and the maximum BHP constraint for an injection segment, respectively, are 725 psi and 2611 psi. The BHP nonlinear constraints are implemented reactively by inputting them directly into the simulator data file. The oil price is $r_{\mathrm{o}} = \$80.0/\mathrm{STB}$ and both the water production and the injection costs are $r_{\mathrm{w}} = r_{\mathrm{winj}} = \$5.0/\mathrm{STB}$. The annual discount rate is 10%.

The initial value for the injection rate of each injection well segment is 1333.0 STB/D and the initial value for the liquid production rate of each production well

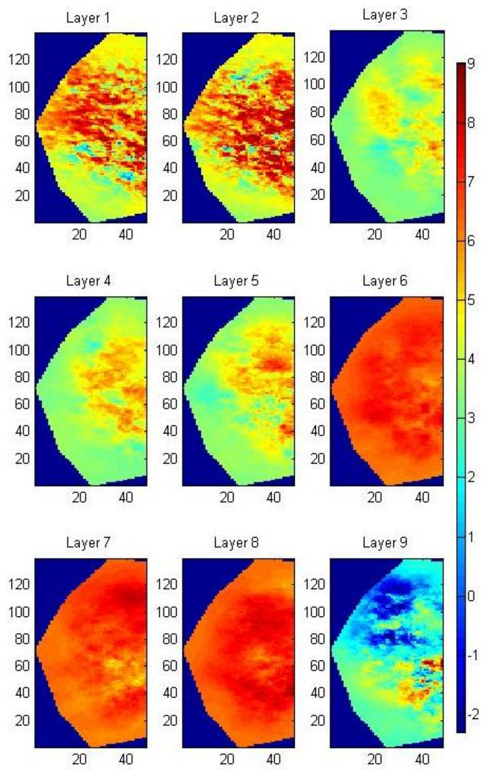


Figure 2.24: $\ln(k_x)$, x-direction log-permeability after history match of period 0-10.

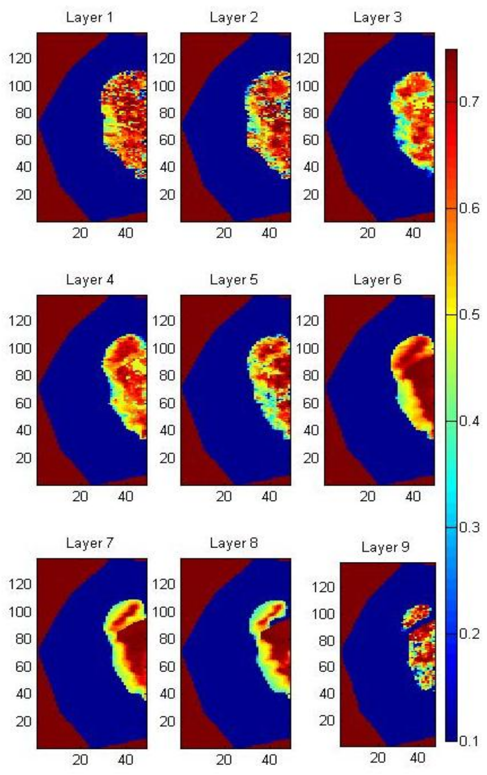


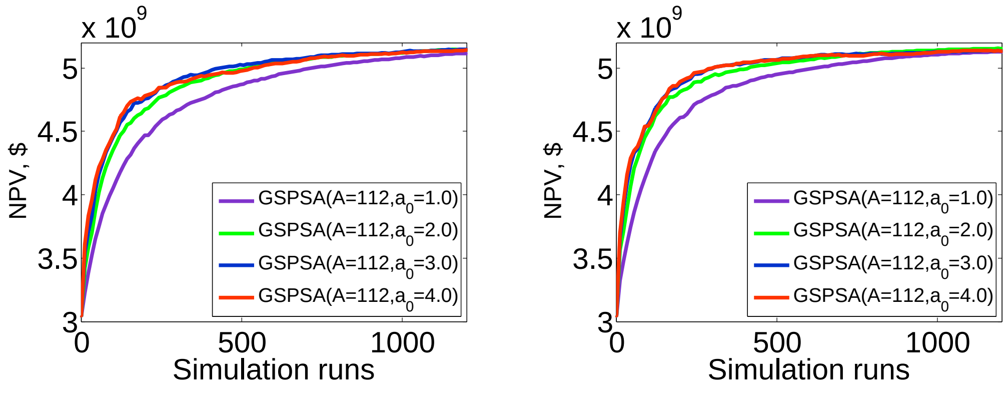
Figure 2.25: Oil Saturation at year 10 before optimization.

segment is 700 STB/D. To solve the bound constraint, we use log-transformation to convert the optimization problem with bound constraints into one without bound constraints. The algorithm will terminate as the iteration reaches the maximum allowable value.

Here, we use the steepest-ascent method where the stochastic gradient is approximated by the average of 10 G-SPSA gradients (Eq. 2.17). The controls of each perturbation are sampled from a Gaussian distribution with a variance pairs of (1,1), of which the first number corresponds to the transformed variable of water injection rate control and the second number corresponds to the transformed variable of liquid production rate, and two correlation lengths of $N_s = 20$ and $N_s = 40$. The normalized search direction (Eq. 2.21) is applied to update the control vector. We set the maximum number of allowable iterations equal to 1120, which is about 1/3 of the total number of control variables, we obtain the value of $A = 112$. We choose $k_{\max} = 1120$ because for field scale problems, it is usually not feasible to do more than 1000 or so reservoir simulation runs. In this example, we set several initial stepsizes of $a_0 = 1.0$, $a_0 = 2.0$, $a_0 = 3.0$, and $a_0 = 4.0$. Corresponding to each value of the initial stepsize, we obtain the value of $a = 17.4$, $a = 35$, $a = 52$, and $a = 69.3$. Based on the experimental results of two previous examples, we know that a good value of $c_{\min}$ is from 0.01 to 0.1. Therefore, we set the value of $c_{\min} = 0.06187$ which gives value of $c = 0.1$.

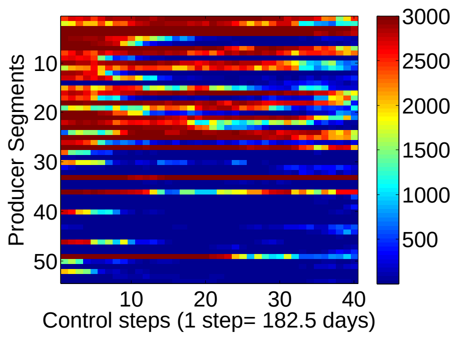
Table 2.20: Compare the performance of G-SPSA algorithms obtained from one optimization runs with one initial random seed when we change the value of N_s and a_0 ($M = 10$, $k_{\max} = 1120$, $A = 112$, and $c_0 = 0.1$; $\sigma^2 = (1, 1)$).

Stepsize a_0	Simulation runs	Final NPV ($N_s = 20$)	Final NPV ($N_s = 40$)
$A = 112, a_0 = 1.0$	1232	5.117×10^9	5.128×10^9
$A = 112, a_0 = 2.0$	1232	5.142×10^9	5.153×10^9
$A = 112, a_0 = 3.0$	1232	5.143×10^9	5.137×10^9
$A = 112, a_0 = 4.0$	1232	5.141×10^9	5.134×10^9

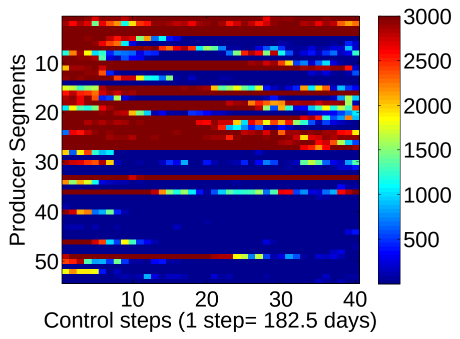


(a) $N_s = 20$ (b) $N_s = 40$

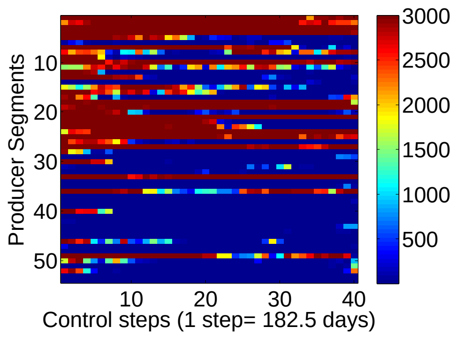
Figure 2.26: NPV versus simulation runs obtained from one optimization runs with one initial random seed when we change the value of N_s and a_0 ($M = 10$, $k_{\max} = 1120$, $A = 112$, and $c_0 = 0.1$; $\sigma^2 = (1, 1)$).



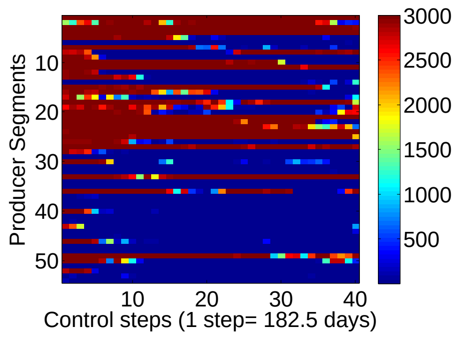
(a) $a_0 = 1.0, N_s = 40$



(b) $a_0 = 2.0, N_s = 40$



(c) $a_0 = 3.0, N_s = 40$

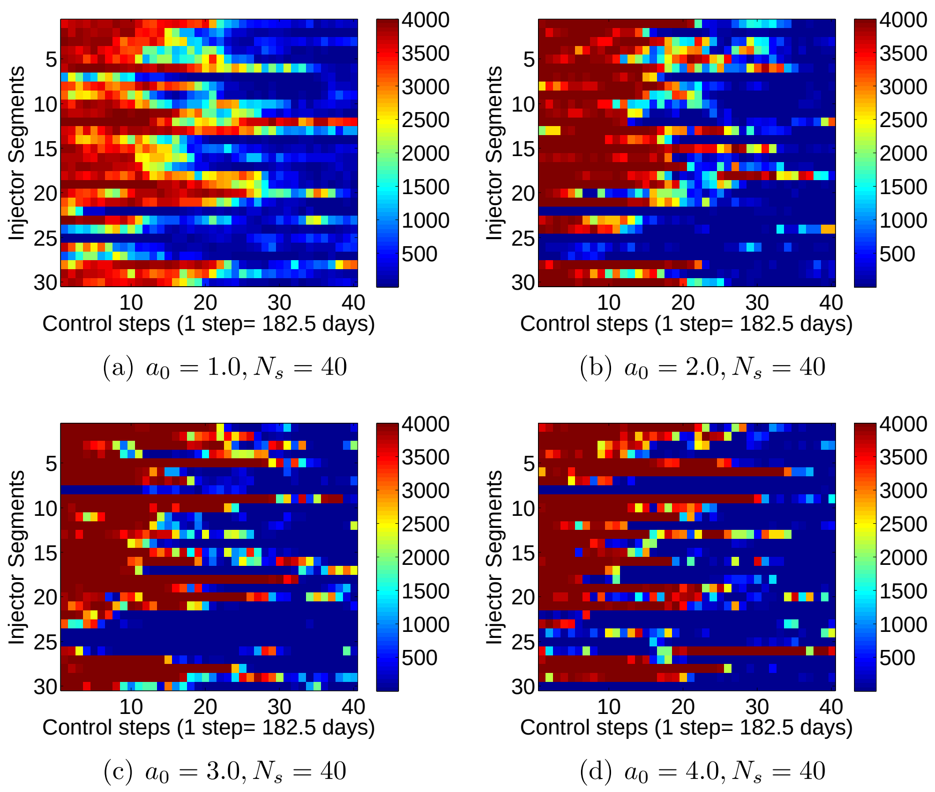


(d) $a_0 = 4.0, N_s = 40$

Figure 2.27: The estimated optimal production well controls obtained from one optimization runs with one initial random seed ($M = 10$, $k_{\max} = 1120$, $A = 112$, and $c_0 = 0.1$; $\sigma^2 = (1, 1)$).

Fig. 2.26 presents NPV versus the number of simulation runs where the G-SPSA algorithm is tested with different values of the initial stepsize and the correlation length. Here, $a_0 = 1$ results in slower convergence for some early iterations but at simulation runs 1232 reaches a similar value of NPV as is obtained with the other choices of a_0. The NPV's in Fig. 2.26 at simulation runs 1232 are given in Table 2.20.

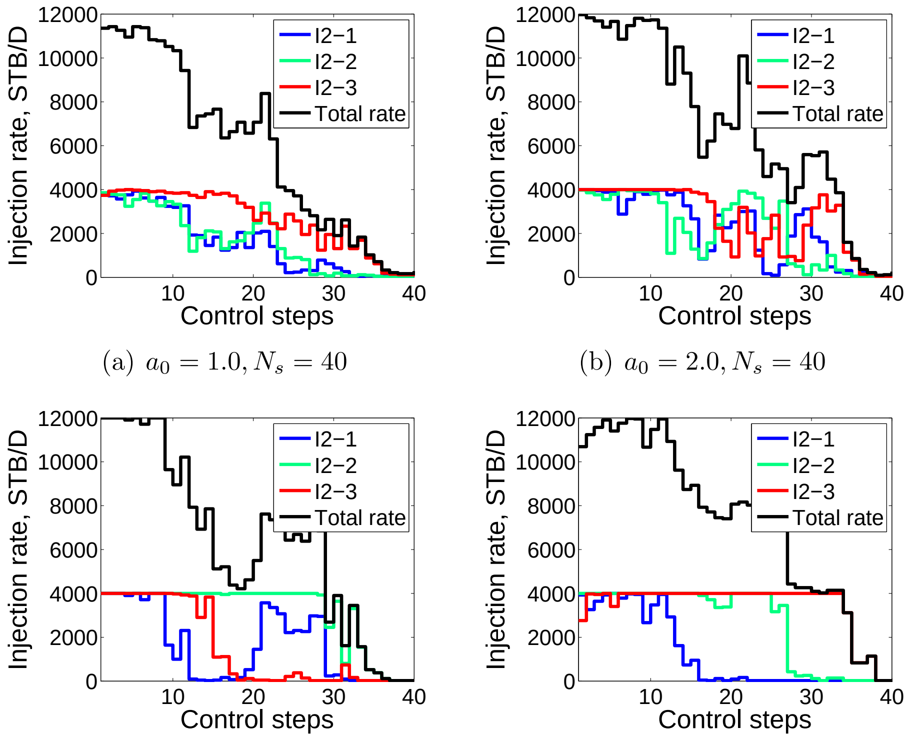
The final controls for the producers and injectors obtained by the G-SPSA algorithm are shown in Figs. 2.28 and 2.27. Note that even though the final NPV's obtained by the G-SPSA algorithm with different SPSA parameters are not very different, the estimates of the optimal controls (Figs. 2.28 and 2.27) are substantially different. This suggests that sometimes there are multiple sets of controls which



(a) $a_0 = 1.0, N_s = 40$ (b) $a_0 = 2.0, N_s = 40$ (c) $a_0 = 3.0, N_s = 40$ (d) $a_0 = 4.0, N_s = 40$

Figure 2.28: The estimated optimal injection well controls obtained from one optimization runs with one initial random seed ($M = 10$, $k_{\max} = 1120$, $A = 112$, and $c_0 = 0.1$; $\sigma^2 = (1, 1)$).

can achieve essentially the same maximum value of NPV. The controls in Fig. 2.27 show high liquid production rates for the first 30 production well segments, which correspond to the first ten producers. From the structure map (Fig. 2.23), these producers are located at the edge of the main fault and are at the farthest locations from the injectors. The wells that are close to the injectors (the segments after 30) produce at much lower liquid rates. Fig. 2.28 show that the water injection rate of almost all the injection segments is high at some of early time control steps. However, there are some injection segments that have a small water injection rate for almost all control steps. The estimates of the optimal total water injection rate and segment rates for Injector 2 as a function of time are shown in Fig. 2.29. Although the final injection controls obtained with different SPSA parameters are very different, the

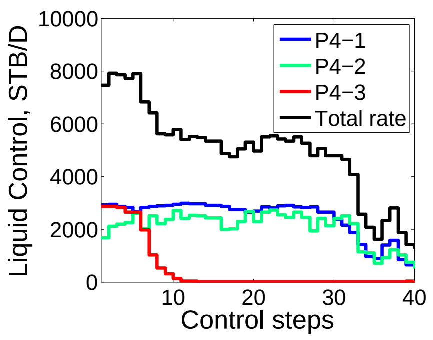


(a) $a_0 = 1.0, N_s = 40$ (b) $a_0 = 2.0, N_s = 40$

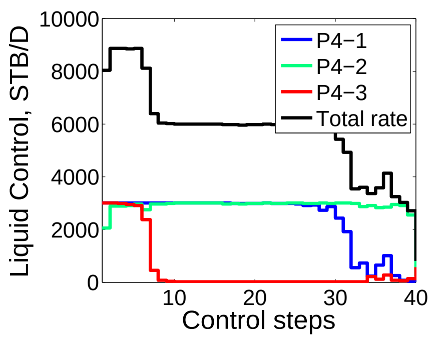
(c) $a_0 = 3.0, N_s = 40$ (d) $a_0 = 4.0, N_s = 40$

Figure 2.29: The estimated optimal well segment injection rates for Injector 2 obtained from one optimization runs with one initial random seed ($M = 10$, $k_{\max} = 1120$, $A = 112$, and $c_0 = 0.1$; $\sigma^2 = (1, 1)$).

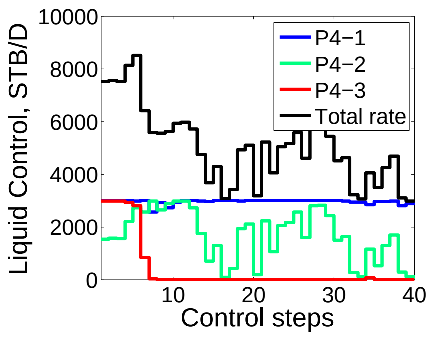
total rate in the injection well is close to the upper bound of 4000 STB/D at the early control steps, and becomes close to zero at the last control step. We see that injection rate controls estimated from G-SPSA are not extremely smooth although we use the full correlation length. Fig. 2.30 shows the optimized rate controls for the three segments of producer P4 from all G-SPSA algorithms with different SPSA parameters. The first segment of this producer, which produces from the top two simulation layers, has a high liquid production rate for almost all control steps. Meanwhile, the liquid production rate in the third segment of this producer, which produces from the simulation layer 6 through 8, is always close to the upper bound at early control steps but decreases to almost zero after the 10^{th} control step, this is the expected result.



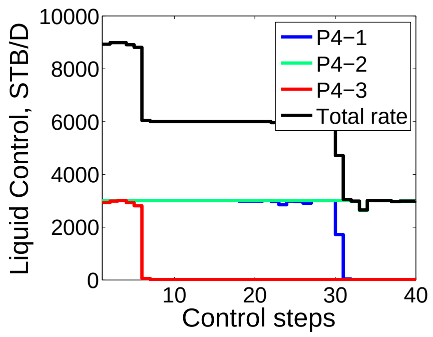
(a) $a_0 = 1.0, N_s = 40$



(b) $a_0 = 2.0, N_s = 40$



(c) $a_0 = 3.0, N_s = 40$



(d) $a_0 = 4.0, N_s = 40$

Figure 2.30: The estimated optimal well segment liquid production rates for Producer 4 obtained from one optimization runs with one initial random seed ($M = 10$, $k_{\max} = 1120$, $A = 112$, and $c_0 = 0.1$; $\sigma^2 = (1, 1)$).

After working with this example in which we wish to maximize the NPV subject to the bound constraints, we can draw some following conclusions:

- The best NPV obtained in this experiment is the same as the NPV $= \$5.161 \times 10^9$ obtained with the adjoint-gradient based algorithm (Chen, 2011). This is an extremely encouraging result as our objective is to obtain an optimization algorithm which does not require an adjoint code to computer the gradient but can obtain a optimal NPV close to the one obtained using an accurate gradient computed from the adjoint method;

- Based on experimental results, we suggest that the value of initial stepsizes from 1.0 to 3.0 all work adequately although in this example the choice $a_0 = 1$ performs slightly worse than $a_0 = 2$ and $a_0 = 3$ especially in term of convergence rate at early iterations (see Fig. 2.26);

- For this example specifying the maximum number of allowable iterations equal to one-third the total number of control variables gave good results at least compared to the NPV obtained using an accurate gradient (Chen, 2011).

CHAPTER 3

AUGMENTED LAGRANGIAN FUNCTION WITH SPSA GRADIENT

The augmented Lagrangian function has proved to be successful in solving nonlinear constrained problems as well as linear constrained problems. This method can alleviate the possibility of ill conditioning by introducing explicit Lagrange multiplier estimates (Conn et al., 1992; Bertsekas, 1995; Nocedal and Wright, 1999). In our implementation, we modify the augmented Lagrangian method used by Chen et al. (2010) who successfully implemented an augmented Lagrangian algorithm to solve the general constrained optimization problem. In the procedure of Chen et al., all gradients required are computed with the adjoint method and the bound constraints were enforced using a standard gradient projection method. The constrained optimization algorithm presented here modifies the Chen et al. procedure in two fundamental ways: (i)gradient projection is not used; instead bounds are enforced using a log-transform, and (ii) the pre-conditioned gradient of the augmented Lagrangian is approximated by an SPSA, and then a preconditioned steepest-ascent algorithm is applied to estimate the optimal well controls subject to the constraints. In our study, a log-transform is applied to each of the original well control variables, and optimization is performed on the transformed variables. The log-transformation ensures that bound constraints are automatically satisfied so that the standard gradient projection technique used to enforce bound constraints is not needed. Although it will not be shown in this work, we have applied the SPSA gradient using the gradient projection method to solve the bound constraints. After doing a lot of experimental computations, we see that the gradient projection method does not always yield reliable results. Thus, we were forced to find a way to avoid the use of gradient projection, which we could also do by transforming the bound constraints to inequality constraints and incorporating them directly into the augmented Lagrangian function. However, using the log-transform to ensure that bound constraints are satisfied has proved to be superior to transforming bound constraints to inequality constraints for all the examples that we have tried.

The augmented Lagrangian method has two loops, outer-loop iterations and inner-loop iterations. Once the controls which maximize the augmented Lagrangian function are obtained in the inner-loop iterations, we update in the outer-loop iteration either the Lagrange multipliers (λ's) or penalty parameter (μ), depending on how well the constraints are satisfied, before moving to the next inner-loop iteration. The above process is repeated until convergence. We discuss the procedure involved in the outer-loop iteration in Section 3.2. In the inner-loop iterations, both λ's and μ are fixed and the augmented Lagrangian function is to be maximized within the bound constraints. We apply a log-transformation to handle the bound constraints. The transformed variables are then adjusted using the SPSA algorithm to maximize the augmented Lagrange function with fixed Lagrange multipliers and a fixed penalty parameter. The inner-loop iteration is discussed in Section 3.3. We then apply our implementation of the SPSA-based augmented Lagrangian method for several specified cases in Section 3.4.

3.1 Introduction to The Augmented Lagrangian Function

The specific production optimization problem considered here pertains to estimating the optimal well controls when waterflooding an oil-reservoir. We assume that there is no gas injection and ignore the income from, or cost of disposal of, produced gas so that the net-present-value (NPV) functional is defined by Eq. 1.1. The NPV defined in Eq. (1.1) is a function of the well control vector u and the dynamic state vector $y = y(u, m)$, which is the vector of variables solved for by the reservoir simulator, i.e., pressures and saturations, m is the reservoir model which is assumed known or at least fixed during production optimization. The well control variables include the water injection rate or the bottomhole pressure (BHP) of the injection wells and the production rate or the BHP of the production wells. The maximization of the NPV in Eq. (1.1) is usually subject to equality, inequality and bound constraints, respectively, given by

$$e_j(u, y) = 0,\ j = 1, .., n_{\mathrm{e}}, \tag{3.1}$$

$$c_j(u, y) \leq 0,\ j = 1, .., n_{\mathrm{ine}}, \tag{3.2}$$

and

$$u_i^{\mathrm{low}} \leq u_i \leq u_i^{\mathrm{up}},\ i = 1, 2, ..., n_{\mathrm{u}}, \tag{3.3}$$

where n_{e}, n_{ine} and n_{u}, respectively, denote the number of equality, inequality and bound constraints. Requiring the field water injection rate to be equal to field liquid production rate is an example of a highly nonlinear equality constraint. Requiring the field and individual water cut to be less than a prescribed value is a nonlinear inequality constraint. The requirement that the sum of the rates at water injection wells be equal to a specified value represents a linear equality constraint. Constraints arise naturally due to the operational limits of the production and injection facilities.

To maximize constrained production optimization problems, we incorporate all equality and inequality constraints into the augmented Lagrangian function (Bertsekas, 1995; Wang and Spall, 2008). Thus the augmented Lagrangian function β is defined as:

$$\begin{aligned}\beta(u,y,\mu,\lambda) = J(u,y) - \sum_{j=1}^{n_\mathrm{e}} \lambda_{\mathrm{e},j} e_j(u,y) - \frac{1}{2\mu}\sum_{j=1}^{n_\mathrm{e}} s_{\mathrm{e},j} e_j^2(u,y) \\ - \sum_{j=1}^{n_\mathrm{ine}} \lambda_{\mathrm{c},j}\left[c_j(u,y)+\nu_j\right] - \frac{1}{2\mu}\sum_{j=1}^{n_\mathrm{ine}} s_{\mathrm{c},j}\left[c_j(u,y)+\nu_j\right]^2,\end{aligned} \tag{3.4}$$

where $J(u,y)$ is the NPV as defined in Eq. 1.1. The ν_j is a non-negative slack variable used to convert the j^{th} inequality constraint ($c_j \leq 0$) into an equality constraint. $\lambda_{\mathrm{e},j}$ denotes the Lagrange multipliers associated with the j^{th} equality constraint, $\lambda_{\mathrm{c},j}$ denotes the Lagrange multipliers associated with the j^{th} inequality constraint. The argument λ in β denotes the vector $\lambda = [\lambda_{\mathrm{e},1}, ..., \lambda_{\mathrm{e},n_\mathrm{e}}, \lambda_{\mathrm{c},1}, ..., \lambda_{\mathrm{c},n_\mathrm{ine}}]^T$. The $s_{\mathrm{e},j}$'s and $s_{\mathrm{c},j}$'s, respectively, denote the scaling factor for the equality and inequality constraints, and μ is the penalty parameter. The convergence rate of the algorithm can be adversely affected by poor scaling. We use the following scaling factors (Chen et al., 2011a):

$$s_{\mathrm{e},j} = \frac{1}{E_j^2},\ j = 1, .., n_\mathrm{e}, \tag{3.5}$$

and

$$s_{\mathrm{c},j} = \frac{1}{C_j^2},\ j = 1, .., n_\mathrm{ine}, \tag{3.6}$$

where E_j and C_j, respectively, are the nonzero constraint values that reflect the magnitude of the j^{th} equality constraint and the j^{th} inequality constraint, respectively (Chen et al., 2011a). For example, if the j^{th} inequality constraint is field water cuts (FWCT) less than 0.9, C_j is set to 0.9. If the j^{th} equality constraint on the total water injection rate is 5000 STB/D, E_j is set to 5000.0. The E_j's and C_j's do not adjust during the optimization process.

In the formulation of Eq. 3.4, the slack variables (ν_j's) are additional adjustable parameters for optimization. Following Nocedal and Wright (1999), we eliminate the slack variables for the inequality constraints in the augmented Lagrangian function. Note that the Lagrangian function of Eq. 3.4 is a concave quadratic function of the slack variable ν_j.

The optimal value for ν_j satisfies

$$\frac{\partial \beta}{\partial \nu_j} = -\lambda_{\mathrm{c},j} - \frac{s_{\mathrm{c},j}}{\mu}[c_j(u,y) + \nu_j] = 0, \tag{3.7}$$

i.e.,

$$\nu_j = -c_j(u,y) - \lambda_{\mathrm{c},j}\frac{\mu}{s_{\mathrm{c},j}}. \tag{3.8}$$

Finally, we find that the optimal values of ν_j in Eq. 3.7 are

$$\nu_j = \max\{-c_j(u,y) - \lambda_{\mathrm{c},j}\frac{\mu}{s_{\mathrm{c},j}}, 0\}, \tag{3.9}$$

where zero includes because we must have $\nu_j \geq 0$.

Using Eq. 3.9 in Eq. 3.4 yields

$$\begin{aligned}\beta(u,y,\mu,\lambda) = J(u,y) - \sum_{j=1}^{n_\mathrm{e}} \lambda_{\mathrm{e},j} e_j(u,y) - \frac{1}{2\mu}\sum_{j=1}^{n_\mathrm{e}} s_{\mathrm{e},j} e_j^2(u,y) \\ - \sum_{j=1}^{n_\mathrm{ine}} \lambda_{\mathrm{c},j}\left[\max\left\{c_j(u,y), -\lambda_{\mathrm{c},j}\frac{\mu}{s_{\mathrm{c},j}}\right\}\right] - \frac{1}{2\mu}\sum_{j=1}^{n_\mathrm{ine}} s_{\mathrm{c},j}\left[\max\left\{c_j(u,y), -\lambda_{\mathrm{c},j}\frac{\mu}{s_{\mathrm{c},j}}\right\}\right]^2.\end{aligned} \tag{3.10}$$

The convergence of the augmented Lagrangian method is guaranteed as the penalty parameter is gradually reduced to zero, almost regardless of the values of the Lagrangian multiplier estimates (Conn et al., 1992). However, it becomes difficult to maximize the augmented Lagrangian function when the penalty parameter is small as small values of μ make the optimization problem less well conditioned. A good choice of Lagrange multipliers can ensure convergence for a fixed penalty parameter provided the vector of current control variables, u_k, is close to a local optimal control vector, u^*. Thus, conceptually, we can decrease the penalty parameter until we are in the neighborhood of u^*, from which point onward the penalty parameter is no longer changed, but rather the Lagrange multiplier estimates adjust to guarantee convergence to u^* and λ^* satisfying the first order Karush−Kuhn−Tucker optimality conditions (Conn et al., 2000).

As in the implementation of the augmented Lagrangian method, the optimization process involved an inner-loop iteration where the the maximization of the augmented Lagrangian is done with fixed λ and μ and an outer-loop iteration where, depending on the magnitude of the violation of the constraints, either all the Lagrange multipliers are modified or the penalty parameter is modified. Normally to deal with the augmented Lagrangian method, the gradient-based method is implemented, and in the inner-loop iteration the augmented Lagrangian function with fixed λ and μ is maximized, subject to the bound constraints, using a gradient projection method. It means that we want to solve the subproblem

$$\begin{aligned} &\max \beta(u, y, \mu, \lambda), \\ &\text{subject to } u_i^{\text{low}} \leq u_i \leq u_i^{\text{up}},\ i = 1, 2, ..., n_{\text{u}}. \end{aligned} \tag{3.11}$$

In our work, we apply the SPSA gradient to generate approximate gradients which are not accurate enough to ensure that the gradient projection method will be reliable. To avoid the gradient projection method, the log-transformation is implemented to remove the bound constraints. There are no bound constraints on the transformed control vector u, in our inner-loop iteration we simply maximize the augmented Lagrangian directly using the SPSA algorithm, i.e., letting λ_ℓ and μ_ℓ denote the values of Lagrange multipliers and penalty parameter at outer loop ℓ at the subsequent inner loop, λ_ℓ and μ_ℓ are held fixed and we maximize $\beta(u, y, \mu_\ell, \lambda_\ell)$.

3.2 Outer-loop Iterations

For the outer-loop iterations, we update either the Lagrange multipliers or the penalty parameter, depending on the magnitude of the violation of the constraints.

When the violation of the constraints is small, we update the Lagrange multipliers but do not change the value of the penalty parameter for the next step. When the violation of the constraints is large, we keep the Lagrange multipliers fixed but decrease the penalty parameter, which allows the algorithm to minimize the violation of the constraints during the next inner-loop step. At the current ℓ^{th} outer-loop iteration, where the inner-loop iteration has converged to u_k, we estimate the violation of the constraints (σ_{c_ν}) by:

$$\sigma_{c_\nu} = \max\left[\max_{1\le j\le n_{\mathrm{e}}}\{s_{\mathrm{e},j}e_j^2(u_k,y_k)\},\max_{1\le j\le n_{\mathrm{ine}}}\{s_{\mathrm{c},j}(\max[c_j(u_k,y_k),0])^2\}\right]. \tag{3.12}$$

Once the inner-loop iteration converges, we check the violation of the constraints using Eq. 3.13, where η^* is the convergence criteria for terminating the algorithm:

$$\sigma_{c_\nu} \le \eta^*. \tag{3.13}$$

If Eq. 3.13 is not satisfied, we go to the next outer-loop iteration and update either the Lagrangian multipliers or the penalty parameter (Conn et al., 1992; Nocedal and Wright, 1999). If Eq. 3.13 is satisfied, we next test convergence conditions for control variables and the objective function using Eqs. 3.14 and 3.15, where ϵ^* and ξ^* are the convergence criteria for terminating the algorithm.

$$\Delta\beta \equiv \frac{|\beta(u_{k+1,\ell})-\beta(u_{k,\ell})|}{\max(|\beta(u_{k+1,\ell})|,1)} \le \epsilon^*, \tag{3.14}$$

$$\Delta u \equiv \frac{\|u_{k+1,\ell}-u_{k,\ell}\|_2}{\max(\|u_{k+1,\ell}\|_2,1)} \le \xi^*. \tag{3.15}$$

If Eqs. 3.14 and 3.15 are satisfied, we stop the algorithm. If Eqs. 3.14 and 3.15 are not satisfied, we go to the next outer-loop iteration and update either the Lagrangian multipliers or the penalty parameter. Note that although the optimization is working in the log-transform domain, Eq. 3.15 is implemented in the original control variables.

To update either the Lagrange multipliers or the penalty parameter in the outer-loop iteration, we need to evaluate magnitude of the violation of the constraints using Eq. 3.12. If the violation of constraints is less than η_ℓ, i.e., if $\sigma_{c_\nu} \leq \eta_\ell$, the tolerance for the violation of the constraints at the outer-loop iteration ℓ, the current value of penalty parameter is adequate to maintain near feasibility of the iterate u_k. In this case, we keep μ_ℓ fixed and update the Lagrange multipliers and tolerance of constraint violation as shown here (Conn et al., 1992):

$$\lambda_{\mathrm{e},j}^{\ell+1} = \lambda_{\mathrm{e},j}^{\ell} + \frac{s_{\mathrm{e},j} e_j(u_{k,\ell}, y_{k,\ell})}{\mu_\ell}, \quad j = 1, 2, ..., n_e, \tag{3.16}$$

$$\lambda_{\mathrm{c},j}^{\ell+1} = \max\{0, \lambda_{\mathrm{c},j}^{\ell} + \frac{s_{\mathrm{c},j} c_j(u_{k,\ell}, y_{k,\ell})}{\mu_\ell}\}, \quad j = 1, 2, ..., n_{ine}, \tag{3.17}$$

$$\mu_{\ell+1} = \mu_\ell, \tag{3.18}$$

and

$$\eta_{\ell+1} = \max(\eta_\ell \mu_{\ell+1}^{\beta_\eta}, \eta^*), \tag{3.19}$$

where β_η is a free parameter. Conn et al. (1992) suggests to using $\beta_\eta = 0.9$ when using an accurate gradient. However, in this study, after doing a lot of experimental computations, we suggest that $\beta_\eta = 0.2$ works fine for a stochastic gradient. Conn et al. (1992) shows that by reducing the tolerance of the violation of the constraints, η_ℓ, the convergence of the augmented Lagrangian function can be guaranteed without driving the penalty parameter to zero. After updating the Lagrange multipliers and tolerance of constraint violation, we decrease the inner-loop convergence criteria for the iteration as

$$\epsilon_{\ell+1} = \max(\epsilon_\ell \mu_{\ell+1}^{\beta_\eta}, \epsilon^*), \tag{3.20}$$

and

$$\xi_{\ell+1} = \max(\xi_\ell \mu_{\ell+1}^{\beta_\eta}, \xi^*). \tag{3.21}$$

Chen et al. (2011a) proposed a new way to estimate the initial Lagrangian multipliers for both equality constraints and inequality constraints using experimental results. The initial Lagrangian multipliers are defined as:

$$\lambda_{\mathrm{e},j}^{0} = \frac{s_{\mathrm{e},j} e_j(u_0, y_0)}{\mu_0}, \quad j = 1, 2, ..., n_e, \tag{3.22}$$

$$\lambda_{\mathrm{c},j}^{0} = \max\{0, \frac{s_{\mathrm{c},j} c_j(u_0, y_0,)}{\mu_0}\}, \quad j = 1, 2, ..., n_{ine}, \tag{3.23}$$

where $e_j(u_0, y_0)$, and $c_j(u_0, y_0)$ are respectively the j^{th} equality constraint and the j^{th} inequality constraint with the initial guess u_0 and $y_0 = y(u_0)$ where as always y is the vector of primary variables solved by the reservoir simulation.

If the violation of the constraints is too severe, i.e., if $\sigma_{c_\nu} > \eta_\ell$, we reduce the penalty parameter and do not change the Lagrange multipliers. By reducing μ_ℓ, we ensure that the augmented Lagrangian function ($\beta(u)$) will place more emphasis on decreasing the violation of the constraints. The following equations are used to update the penalty parameter and tolerance of constraint violation:

$$\mu_{\ell+1} = \tau \mu_\ell, \tag{3.24}$$

and

$$\eta_{\ell+1} = \max(\eta_0 \mu_{\ell+1}^{\alpha_\eta}, \eta^*), \tag{3.25}$$

where the value of τ can be from 0.1 to 0.5 and α_η is a free parameter. After doing several experimental computations, we suggest that $\alpha_\eta = 0.1$ works well for a stochastic gradient. The value of α_η is the same as recommendation of Conn et al. (1992). The convergence criteria for the next outer-loop iteration are reset according to the following equations (Conn et al., 1992):

$$\epsilon_{\ell+1} = \max(\epsilon_0 \mu_{\ell+1}^{\alpha_\eta}, \epsilon^*), \tag{3.26}$$

and

$$\xi_{\ell+1} = \max(\xi_0 \mu_{\ell+1}^{\alpha_\eta}, \xi^*). \tag{3.27}$$

How to choose the initial guess for the penalty parameter μ_0 is a challenging task and the choice can have an impact on the performance of the algorithm. Chen (2011) used experimental computations to estimate a good initial value of the penalty parameter. However, in our work, by using the scaling factors (see Eqs. 3.5 and 3.6) and the initial Lagrange multipliers (see Eqs. 3.22 and 3.23), we expect the violation of constraints to be on the order of 1 so $1/\mu_0$ times the violation should be roughly in the order of the expected value of NPV. We suspect that ideally the value of $1/\mu_0$ should be on the order of 1/10 the expected value of J.

3.3 Inner-loop Iterations

In the inner-loop iterations, the transformed variables are adjusted using the SPSA algorithm to maximize the augmented Lagrange function with fixed Lagrange multipliers and a fixed penalty parameter. The SPSA gradient is applied to approximate the gradient of the augmented Lagrangian function. To do this, the transformed variables are perturbed using a Bernoulli distribution or a Gaussian distribution sampled from a normal distribution with mean equal to the no-dimensional zero vector and covariance matrix, C_U (see Eq. 2.15). The normalized search direction is applied to estimate the new transformed variables (see Eq. 2.21). In inner-loop iteration, the value of step size and pertubation size are respectively updated using Eqs. 2.19 and 2.20. We do not terminate the inner-loop iteration using the magnitude of the augmented Lagrangian function's gradient as the gradient projection method did;

instead the following convergence criteria are implemented:

$$\Delta\beta \equiv \frac{|\beta(u_{k+1,\ell}) - \beta(u_{k,\ell})|}{\max(|\beta(u_{k+1,\ell})|, 1)} < \epsilon_\ell, \tag{3.28}$$

and

$$\Delta u \equiv \frac{\|u_{k+1,\ell} - u_{k,\ell}\|_2}{\max(\|u_{k+1,\ell}\|_2, 1)} < \xi_\ell, \tag{3.29}$$

where k denotes the inner-loop iteration index, ℓ is the outer-loop iteration index, and ϵ_ℓ and ξ_ℓ are the inner-loop convergence criteria. Note that in the inner-loop iteration, the value of ℓ is fixed. We also note that although the optimization is working in the log-transform domain, Eq. 3.29 keeps working in the original control variables.

3.4 Case Study

3.4.1 Production optimization with equality and bound constraints

We consider a synthetic water flooding case that was considered previously by Brouwer and Jansen (2004) and Sarma et al. (2008). We implement production optimization with the true model represented by a horizontal reservoir with a uniform grid system, $45 \times 45 \times 1$ with $\Delta x = \Delta y = 450$ ft. The thickness of the reservoir is 10 ft. The fluid system is two-phase flow of water and oil. The log-permeability field is shown in Fig. 3.1 with two high permeability channels indicated in dark red color. The porosity is homogeneous throughout the reservoir ($\phi = 0.25$). The initial reservoir pressure is 5800 psi and the connate water saturation and residual oil saturation are both equal to 0.1. As done by Brouwer and Jansen (2004) and Sarma et al. (2008), we use 45 injection wells at the left side of the system to simulate a smart horizontal injection well with 45 injection segments and 45 production wells at the right side of the reservoir to simulate a smart horizontal production well with 45 production segments. The anticipated water flooding project life is 960 days and we set each control step equal to 190 days. Thus, we have 5 control steps and the total number of control variables is equal to $(45 + 45) \times 5 = 450$.

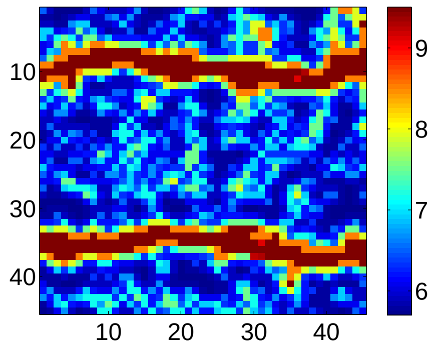


Figure 3.1: ln(k) distribution.

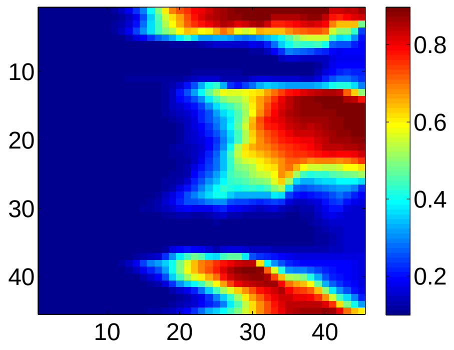


Figure 3.2: Final oil saturation with reactive control.

During production optimization, the injection segments are placed under rate control with a lower bound of 0 STB/D and an upper bound of 500 STB/D. The production segments are placed under bottom hole pressure (BHP) control with a lower bound of 3500 psi and an upper bound of 6000 psi. Moreover, the total field water injection rate (FWIR) at each control step should be equal to 2700 STB/D. It means that the equality constraints in Eq. 1.2 are

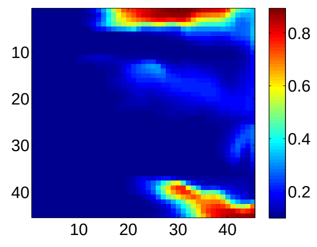
$$e_j = \sum_{i=1}^{45} q_{winj,i}^{j} - 2700 \text{ (STB/D)} = 0, \; j = 1, \ldots, 5, \tag{3.30}$$

where i denotes the injection well index, and j denotes the equality constraint index which is the same as control step index. Therefore, we have 5 linear equality constraints which are added to the NPV functional to form the augmented Lagrangian function in Eq. 3.10. In this work, we use the same cost data as in Brouwer and Jansen (2004). The oil price is set at \$12.7/STB, the water injection rate cost at \$0/BBL , the water production cost at \$3.18/STB, and the annual discount rate is 10%.

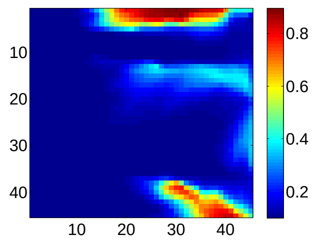
The initial guess for control variables is 60 STB/D for each injection segment and 5750 psi for each production segment at all control steps. We set the scaling factors for equality constraints $s_{e,j} = 1/E_j^2$ in which E_j is set to 2700 STB/D for all j's. The initial Lagrange multipliers are calculated by using Eq. 3.22, and the initial penalty parameter, μ_0, is set to 10^{-7}. Note that with this choice, we expect the violation of constraints to be on the order of 1 so $1/\mu_0$ times the violation should be roughly in the order of 10^7. We suspect that ideally this 10^7 value should be on the order of 1/10 the value of J. The initial values of the convergence tolerances in Eqs. 3.28 and 3.29, respectively, are set to $\Delta\beta_{k,\ell} \leq \epsilon_0 = 0.001$, $\Delta u_{k,\ell} \leq \xi_0 = 0.01$, and these values decrease very slowly from outer-loop iteration to outer-loop iteration. The initial tolerance of the violation of the constraints, η_0, is set to 0.0025. The algorithm terminates when $\Delta\beta_{k,\ell} \leq \epsilon^* = 0.6\epsilon_0$, $\Delta u_{k,\ell} \leq \xi^* = 0.6\xi_0$ and $\sigma_{c_\nu} \leq \eta^* = 0.01\eta_0$.

Here, we use the steepest-ascent method where the stochastic gradient is approximated by the average of five SPSA gradients (Eq. 2.17) calculated by one-sided simultaneous perturbation (Eqs. 2.2 and 2.4). The normalized search direction (Eq. 2.21) is used to update the control vector. For the G-SPSA algorithm, the controls of each perturbation are sampled from a Gaussian distribution with the correlation length of $N_s = 2$ and the variance of transformed variable of BHP control and transformed variable of water injection rate control, respectively, of 25 and 5. The choice of 25 and 5 for variance is a bit strange based on our early use of all variances equal to 1.0. However, with the choice of $c_{\min} = 0.055$ which gives $c = 0.1$, we found these variances gave a better result than is obtained with all variances equal to 1. By setting the maximum number of allowable iterations equal to 450 which is identical to the total number of control variables, we obtain the value of A equal to 45. We set $a_0 = 1.0$ which gives $a = 14.1$.

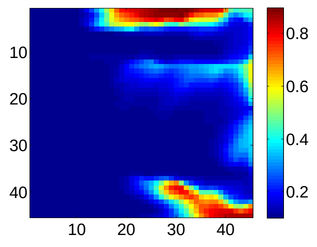
To show how production optimization improves the recovery, we first set up a reference case, which is run with reactive controls. Reactive control refers to the procedure whereby we simply shut in well segment whenever the cost of disposing of the produced water exceeds the oil revenue, i.e., when the water oil ratio (WOR) at the well segment exceeds $12.7/3.18 = 4.0$. As shown in Fig. 3.2, we obtain a very poor sweep efficiency and poor oil recovery for the reactive control case.



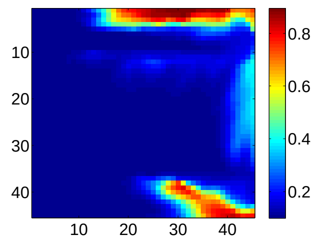
(a) B-SPSA



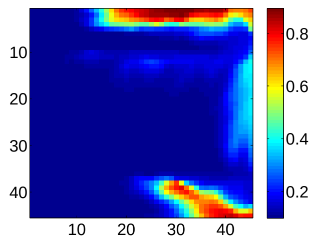
(b) ComB-SPSA1



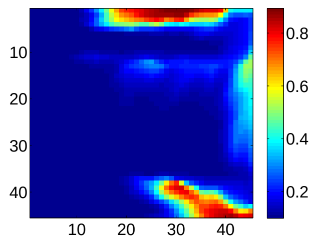
(c) ComB-SPSA2



(d) G-SPSA



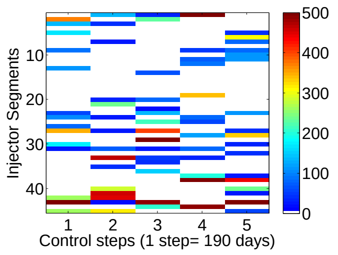
(e) ComG-SPSA1



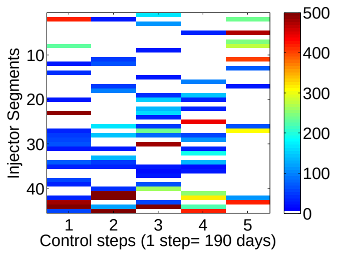
(f) ComG-SPSA2

Figure 3.3: Remaining oil saturation distribution obtained from one optimization run with one initial random seed ($M = 5$, $k_{\max} = 450$, $A = 45$, $a_0 = 1$, and $c_0 = 0.1$; $N_s = 2$ and $\sigma^2 = (25, 5)$; $\mu_0 = 10^{-7}$).

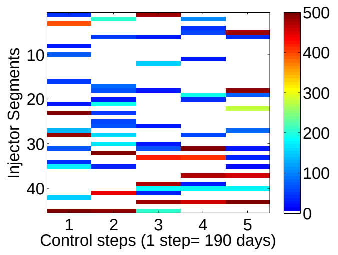
The final oil saturation distribution after optimization from various SPSA algorithms is shown in Fig. 3.3. Compared to the reactive control case (Fig. 3.2), the SPSA optimal controls yield a much better sweep efficiency and oil recovery. The low permeability region between two high permeability channels is much better swept. The reasons leading to a better sweep efficiency can be easily explained by analyzing the optimized controls, the water injection rate for injection segments and the BHP's for production segments. Fig. 3.4 presents the final control variables of water injection rate obtained from SPSA algorithms. The y-axis in this figure corresponds to the 45 water injection segments and the x-axis corresponds to the 5 control steps. The color scale corresponds to injection rates of the segments, with



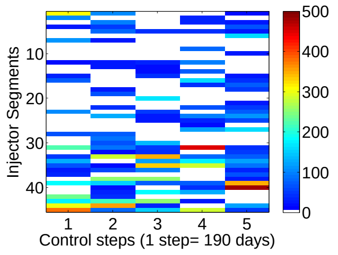
(a) B-SPSA



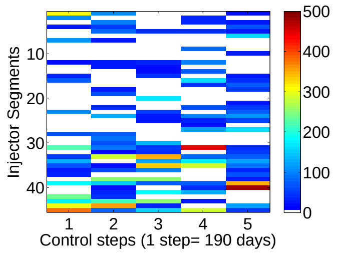
(b) ComB-SPSA1



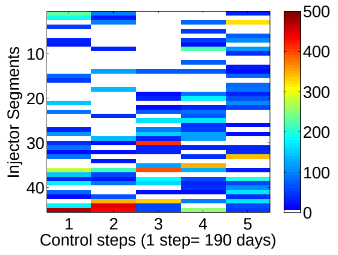
(c) ComB-SPSA2



(d) G-SPSA



(e) ComG-SPSA1

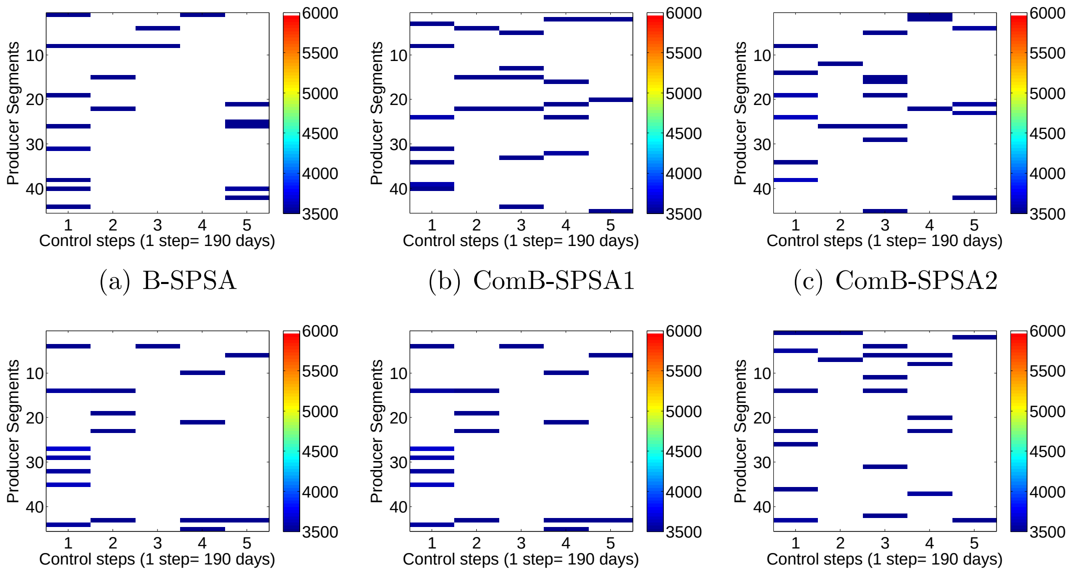


(f) ComG-SPSA2

Figure 3.4: The estimated optimal water injection rate well controls obtained from one optimization run with one initial random seed ($M = 5$, $k_{\max} = 450$, $A = 45$, $a_0 = 1$, and $c_0 = 0.1$; $N_s = 2$ and $\sigma^2 = (25, 5)$; $\mu_0 = 10^{-7}$).

white representing a rate of approximately zero and red representing the highest rate. It is clear that the water injector segments completed in or near the top high permeability streaks are shut down at early times. The results that are clearest to interpret are those obtained from G-SPSA, ComG-SPSA1, and ComG-SPSA2. In these cases, we see that the injection rate into the top high permeability channel of Fig. 3.1 is zero and more water is injected into the lower part of reservoir so the sweep tends to be from bottom to top. The estimated pressure controls at the production segment (Fig. 3.5) are somewhat surprising as only a few production segments are active and produce at a pressure fairly close to the minimal allowable pressure of 3500 psi. In Figs. 3.5 and 3.4, white is used to indicate that a well is shut in.

Fig. 3.6 presents the violation of FWIR constraints versus the number of simulation runs. In each sub-figure of Fig. 3.6, we have five curves of FWIR-2700 corresponding to five control steps. Whenever each of those curves is not equal to 0, the constraint is violated. Fig. 3.6 shows that for the B-SPSA algorithms, the



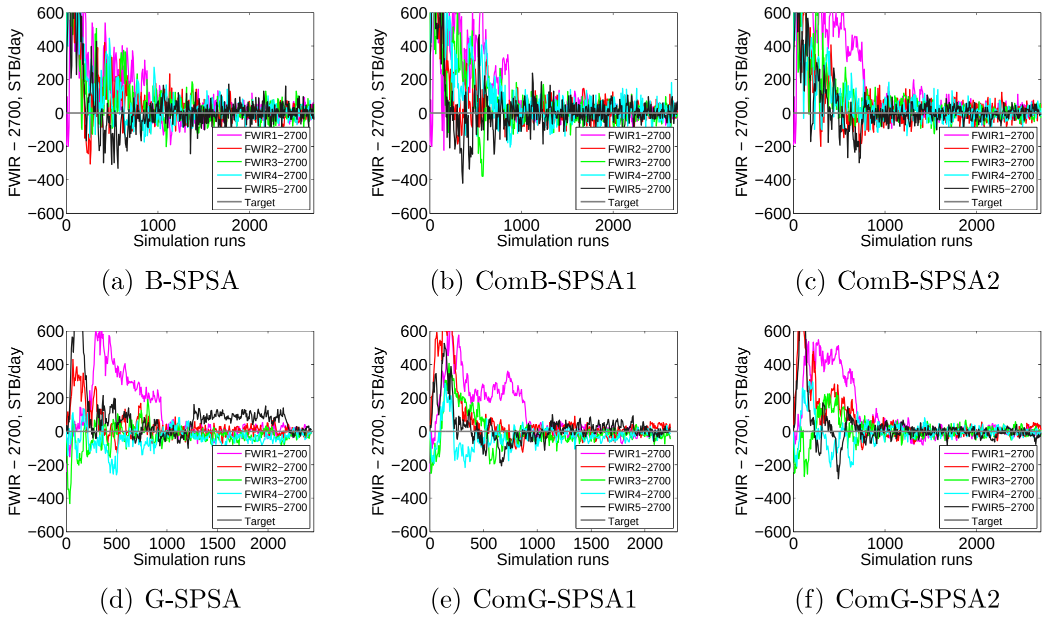
(a) B-SPSA (b) ComB-SPSA1 (c) ComB-SPSA2

(d) G-SPSA (e) ComG-SPSA1 (f) ComG-SPSA2

Figure 3.5: The estimated optimal BHP well controls obtained from one optimization run with one initial random seed($M = 5$, $k_{\max} = 450$, $A = 45$, $a_0 = 1$, and $c_0 = 0.1$; $N_s = 2$ and $\sigma^2 = (25, 5)$; $\mu_0 = 10^{-7}$).

constraints are still violated even when the iteration index reaches to the maximum number of allowable iterations which is 450. However, the violation of constraints obtaining from the G-SPSA algorithms becomes close to zero as simulation runs increase. The G-SPSA algorithm terminates when $\Delta\beta_{k,\ell} \leq \epsilon^* = 0.0006$ and $\Delta u_{k,\ell} \leq \xi^* = 0.006$. At the final iteration, the violation of constraints becomes less than the final tolerance of constraint violation, i.e, $\sigma_{c_\nu} \leq 0.000025$. It means that the violation of the constraints is satisfied within the specified tolerance at convergence.

The performance of the SPSA algorithms is presented in Table 3.1. Even though the final three NPV's based on using Bernoulli perturbations were only slightly lower than those obtained using Gaussian perturbations, it is important to note the optimization algorithm did not converge when the Bernoulli perturbations were used because the violation of constraints was greater than the specified tolerance. In B-SPSA, the violation of the constraints, σ_{c_ν}, is around 0.0002 after 450 iterations (2700 reservoir simulation runs). Note that this σ_{c_ν} value is greater than



(a) B-SPSA (b) ComB-SPSA1 (c) ComB-SPSA2

(d) G-SPSA (e) ComG-SPSA1 (f) ComG-SPSA2

Figure 3.6: Constraint violation versus simulation run obtained from one optimization run with one initial random seed ($M = 5$, $k_{\max} = 450$, $A = 45$, $a_0 = 1$, and $c_0 = 0.1$; $N_s = 2$ and $\sigma^2 = (25, 5)$; $\mu_0 = 10^{-7}$).

the convergence tolerance of constraint violation, $\eta^* = 0.000025$. As the physical problem suggests that it is possible to obtain a higher NPV if constraints are not imposed, the Bernoulli results are not meaningful. In addition, ComG-SPSA2 did not converge and was terminated because the maximum number (2700) of allowable simulation runs was reached. The ComG-SPSA1 algorithm did, however, satisfy the convergence tolerance of constrain violation at termination. In Table 3.1, N_μ denotes the number of times the penalty parameter was adjusted at an outer-loop iteration and N_λ denotes the number of times the Lagrange multipliers were adjusted at an outer-loop iteration. $N_\mu + N_\lambda$ is equal to the total number of outer-loop iterations. As the results of Table 3.1 indicate when perturbations based on the Bernoulli distributions the Lagrange multipliers were never changed, only the penalty parameter was updated at the outer-loop iterations. The failure to update the Lagrangian multipliers means the algorithm is reduced to a penalty method rather than an augmented Lagrangian procedure. The ComG-SPSA1 algorithm gives the best result

Table 3.1: The performance of different algorithms obtained from one optimization with one initial random seed (N_μ: Number of μ updated; N_λ: Number of λ updated; $M = 5$, $k_{\max} = 450$, $A = 45$, $a_0 = 1$, and $c_0 = 0.1$; $N_s = 2$ and $\sigma^2 = (25, 5)$; $\mu_0 = 10^{-7}$)

Algorithm	Simu-lations	Final NPV,\$	Outer -loop	N_μ	N_λ	Conver-gence
B-SPSA	2700	2.034×10^7	3	3	0	No
ComB-SPSA1	2700	2.017×10^7	3	3	0	No
ComB-SPSA2	2700	1.959×10^7	4	4	0	No
G-SPSA ($N_s = 2$)	2430	2.049×10^7	4	2	2	Yes
ComG-SPSA1 ($N_s = 2$)	2250	2.056×10^7	5	3	2	Yes
ComG-SPSA2 ($N_s = 2$)	2700	2.014×10^7	6	4	2	No

of NPV = \$$2.056 \times 10^7$ but it is essentially equal to the NPV= \$$2.049 \times 10^7$ from G-SPSA required 2430 simulation runs to reach convergence compared to only 2250 simulation runs with ComG-SPSA1.

In summary, the results of the G-SPSA-based augmented Lagrangian algorithm performanced well for this example but B-SPSA did not. The NPV = \$$2.049 \times 10^7$ is similar to the value of NPV= \$$2.01 \times 10^7$ obtained by Brouwer and Jansen (2004) who used gradient-based method and the value of NPV = \$$2.067 \times 10^7$ obtained by Chen (2011) who used the adjoint method and the standard gradient projection method to solve the bound constraints.

3.4.2 Production optimization with inequality and bound constraints

Here, we use the same reservoir model and well data as in Subsection 3.4.1. During production optimization, the injection segments are placed under rate control with a lower bound of 0 STB/D and an upper bound of 500 STB/D. The production segments are placed under bottom hole pressure (BHP) control with a lower bound of 3500 psi and an upper bound of 6000 psi. However, the total water injection rate (FWIR) of all injection wells at each control step is changed from an equality or an inequality constraint, i.e., we require that the FWIR less than or equal to 2700

STB/D. It means that the inequality constraints in Eq. 1.3 can be written as

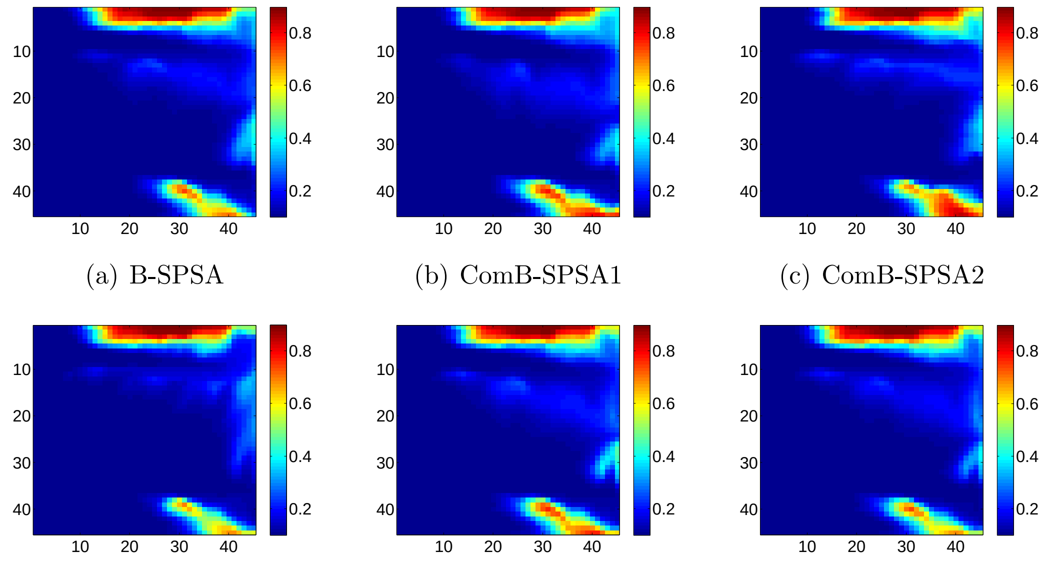
$$c_j = \sum_{i=1}^{45} q_{winj,i}^{j} - 2700 \leq 0 \text{ (STB/D)}, \; j = 1, \ldots, 5, \tag{3.31}$$

where i denotes the injection well index, and j denotes the inequality constraint index as well as control step index. Therefore, we have five linear inequality constraints which are added to the NPV functional to form the augmented Lagrangian function in Eq. 3.4. We use the same cost data as in Subsection 3.4.1, i.e., the oil price is set at \$12.7/STB, the water injection rate cost is set at \$0/BBL, the water production cost is set at \$3.18/STB, and the annual discount rate is 10%.

To maximize the NPV subject to linear inequality constraints, we compute an average of five SPSA gradients calculated by Eqs. 2.2 and 2.4. Then, we use the normalized search direction (Eq. 2.21) to update the control vector. Control variables are sampled from Bernoulli distribution and Gaussian distribution which has a variance pair of (1,1), of which the first number corresponds to the transformed variable of BHP control and the second number corresponds to the transformed variable of water injection rate control, and a correlation length of $N_s = 2$ (Eq. 2.15). The value of α and γ in those equations is again $\alpha = 0.602$ and $\gamma = 0.101$. By setting the maximum number of allowable iterations equal to 450 which is identical to the number of control variables, we obtain the value of A equal to 45. By setting $a_0 = 1.0$, we obtain $a = 14.1$. We set c_{min} equal to 0.055 which gives $c = 0.1$.

The initial guess for control variables is 60 STB/D for each injection segment and 5750 psi for each production segment at all control steps (initial guess 1). We set the scaling factors for inequality constraints $s_{c_j} = 1/C_j$. For field water injection rate constraints, C_j is set equal to 2700 STB/D for all j's. The initial penalty parameter is set to 10^{-7}. As discussed previously, we suspect that ideally this 10^7 value should be on the order of 1/10 the expected value of J (NPV). The initial

Lagrange multipliers are estimated by using Eq. 3.23. The initial values of the convergence tolerances in Eqs. 3.28 and 3.29, respectively, are set to $\Delta\beta_{k,\ell} \leq \epsilon_0 = 0.001$ and $\Delta u_{k,\ell} \leq \xi_0 = 0.01$, and these values decrease very slowly from outer-loop iteration to outer-loop iteration. The initial tolerance of the violation of the constraints is equal to 0.0025, i.e., $\eta^0 = 0.0025$. The algorithm terminates when $\Delta\beta_{k,\ell} \leq \epsilon^* = 0.6\epsilon_0$, $\Delta u_{k,\ell} \leq \xi^* = 0.6\xi_0$ and the violation of the constraints is less than or equal to 0.000025, i.e, $\sigma_{c_\nu} \leq \eta^* = 0.01\eta_0$.



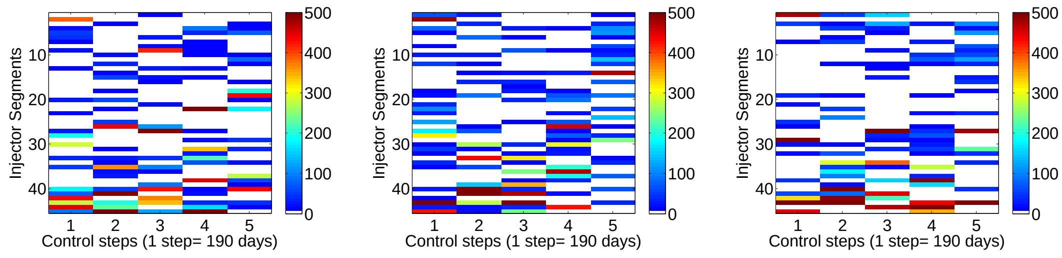
(a) B-SPSA (b) ComB-SPSA1 (c) ComB-SPSA2

(d) G-SPSA (e) ComG-SPSA1 (f) ComG-SPSA2

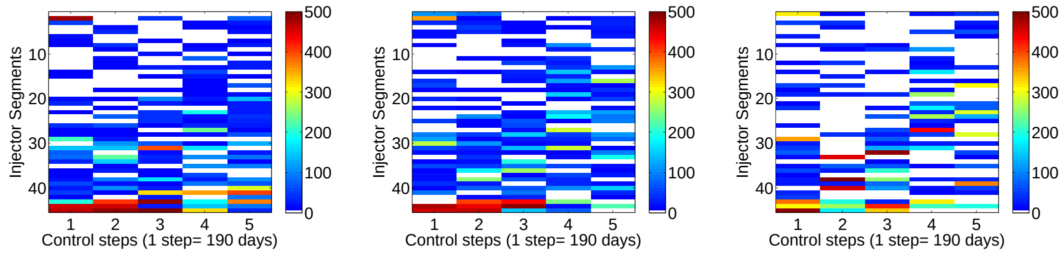
Figure 3.7: Remaining oil saturation distribution obtained from one optimization run with one initial random seed ($M = 5$, $k_{\max} = 450$, $A = 45$, $a_0 = 1$, and $c_0 = 0.1$; $N_s = 2$ and $\sigma^2 = (1, 1)$; $\mu_0 = 10^{-7}$; Initial guess 1).

The final oil saturation distribution using the final optimal controls obtained from various SPSA algorithms with initial guess 1 is shown in Fig. 3.7. Compared to the reactive control case (Fig. 3.2), the SPSA optimal controls yield a much better sweep efficiency and oil recovery. The low permeability region between two high permeability channels is much better swept. The reasons leading to the better sweep in the optimization case can be easily explained by analyzing the optimized controls, the water injection rate for injection wells and BHP's for producer wells. Fig. 3.8

presents the final control variables of water injection rate obtained from the SPSA algorithms. The y-axis in this figure corresponds to the 45 water injection segments and the x-axis corresponds to the 5 control steps. The color scale corresponds to injection rates of the segments, with white representing a rate very close to zero and red representing highest rate of 500 STB/D. It is clear that the water injector segments completed in or near the top high permeability channels are shut-in at early times, so that similar to the case with a equality constraint for the FWIR, the sweep from bottom to the top. The estimated pressure controls at the producer (Fig. 3.9) are somewhat surprising as only a few producers are active, however these producers are at a pressure fairly close to the minimal allowable pressure of 3500 psi as expected. In Figs. 3.9, white is used to indicate that the well is shut in.



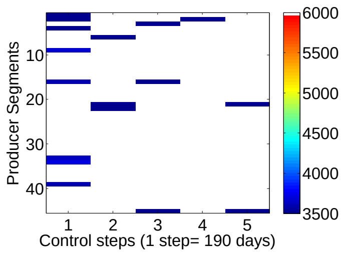
(a) B-SPSA (b) ComB-SPSA1 (c) ComB-SPSA2



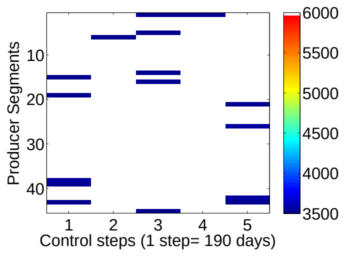
(d) GSPSA (e) ComGSPSA1 (f) ComGSPSA2

Figure 3.8: The estimated optimal water injection rate well controls obtained from one optimization run with one initial random seed ($M = 5$, $k_{\max} = 450$, $A = 45$, $a_0 = 1$, and $c_0 = 0.1$; $N_s = 2$ and $\sigma^2 = (1, 1)$; $\mu_0 = 10^{-7}$; Initial guess 1).

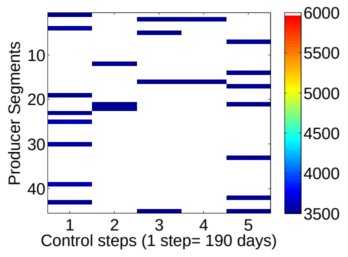
Fig. 3.10 presents the relationship between the violation of the field water injection rate constraints and the number of simulation runs using initial guess 1. Note



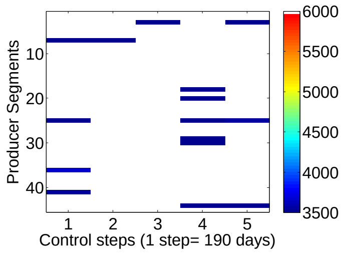
(a) B-SPSA



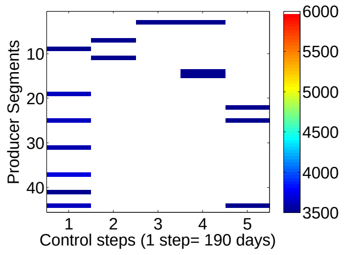
(b) ComB-SPSA1



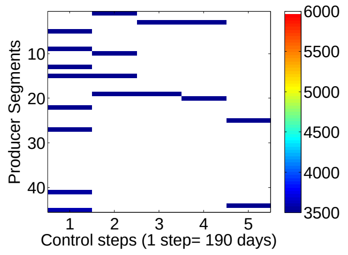
(c) ComB-SPSA2



(d) GSPSA



(e) ComGSPSA1

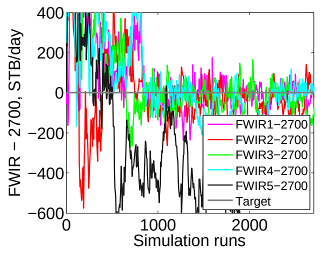


(f) ComGSPSA2

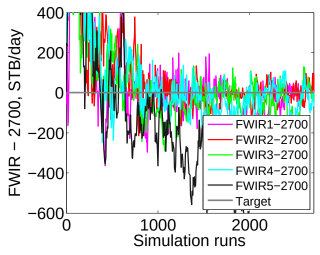
Figure 3.9: The estimated optimal BHP well controls obtained from one optimization run with one initial random seed ($M = 5$, $k_{\max} = 450$, $A = 45$, $a_0 = 1$, and $c_0 = 0.1$; $N_s = 2$ and $\sigma^2 = (1, 1)$; $\mu_0 = 10^{-7}$; Initial guess 1).

that we plot the field water injection rate (FWIR) minus 2700 STB/D versus the number of simulation runs, so whenever FWIR $-$ 2700 (STB/D) > 0 the constraint is violated. The five curves for FWIR $-$ 2700 (STB/D) correspond to the five control steps. As the B-SPSA algorithms approach the maximum allowable iteration numbers, the violation of the constraints is still not satisfied. Meanwhile, the G-SPSA algorithms terminate when $\Delta\beta_{k,\ell} \leq 0.0006$ and $\Delta u_{k,\ell} \leq 0.006$, and the constraints of FWIR is less than or equal to 25 STB/D (0.5% of the constraint FWIR value 2700 STB/D), i.e., satisfy the specified tolerance on the violation of the constraints. From the early simulation runs to around the 1000^{th} simulation run, constraints are significantly violated. However, after several outer-loop iterations, as the penalty parameter reduces the constraint violation also decreases. At the final simulation run, all constraints become less than 0 STB/D (see Fig. 3.10).

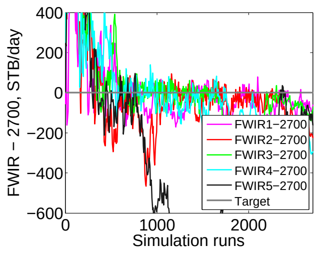
The performance of the SPSA algorithms is presented in Table 3.2. Generally, all G-SPSA algorithms obtain convergence after three or four outer-loop iterations



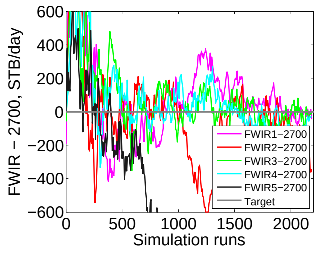
(a) B-SPSA



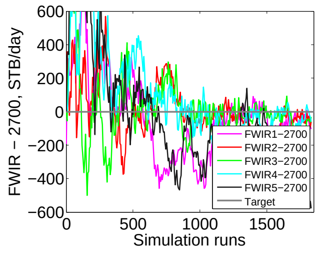
(b) ComB-SPSA1



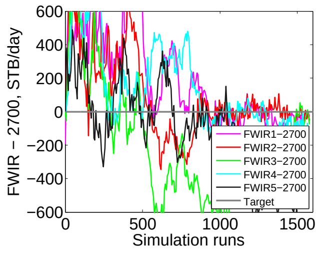
(c) ComB-SPSA2



(d) G-SPSA



(e) ComG-SPSA1



(f) ComG-SPSA2

Figure 3.10: Constraint violation versus simulation runs obtained from one optimization run with one initial random seed ($M = 5$, $k_{\max} = 450$, $A = 45$, $a_0 = 1$, and $c_0 = 0.1$; $N_s = 2$ and $\sigma^2 = (1, 1)$; $\mu_0 = 10^{-7}$; Initial guess 1).

and do at least one update of the Lagrange multipliers. The ComG-SPSA2 algorithm gives the best result in term of NPV $= \$2.074 \times 10^7$ with the smallest number of reservoir simulation runs, 1578, but our basic algorithm, G-SPSA still performs adequately. Table 3.3 presents the results obtained from a variance pair of (25,5) and (1,1). Similar to the equality constraint case, the results for the variance (1,1) are slightly worse than those for (25,5).

Table. 3.4 presents the performance of the G-SPSA algorithm with several different initial penalty parameters. Whenever the constraint violation is not satisfied, the value of penalty parameter will be reduced by 0.1 in the next outer-loop iterations. As long as the value of the initial penalty parameter is less than or equal to 10^{-4}, the algorithm converges. Overall $\mu_0 = 10^{-7}$ gives the best result. The bigger value of the initial penalty parameter is, the greater number of outer loop-iterations is. As we require several reduction in μ before we do not significantly violate the

Table 3.2: The performance of different algorithms obtained from one optimization with one initial random seed ($\mu^0 = 10^{-7}$; Var$= 1, 1$; N_μ: Number of μ updated; N_λ: Number of λ updated; ($M = 5$, $k_{\max} = 450$, $A = 45$, $a_0 = 1$, and $c_0 = 0.1$; $N_s = 2$ and $\sigma^2 = (1, 1)$; $\mu_0 = 10^{-7}$; Initial guess 1).

Algorithm	Simu-lations	Final NPV,\$	Outer -loop	N_μ	N_λ	Conver-gence
B-SPSA	2700	2.089×10^7	4	4	0	No
ComB-SPSA1	2700	2.087×10^7	4	4	0	No
ComB-SPSA2	2700	2.052×10^7	4	4	0	No
GSPSA ($N_s = 2$)	2184	2.062×10^7	4	2	2	Yes
ComGSPSA1 ($N_s = 2$)	1830	2.047×10^7	3	2	1	Yes
ComGSPSA2 ($N_s = 2$)	1578	2.074×10^7	3	2	1	Yes

Table 3.3: The performance of different algorithms obtained from one optimization with one initial random seed when we change the value of variances (N_μ: Number of μ updated; N_λ: Number of λ updated; $M = 5$, $k_{\max} = 450$, $A = 45$, $a_0 = 1$, and $c_0 = 0.1$; $N_s = 2$; $\mu_0 = 10^{-7}$; Initial guess 1).

Algorithm	Simu-lations	Final NPV,\$	Outer -loop	N_μ	N_λ	Conver-gence
G-SPSA ($\sigma^2 = 1, 1$)	2184	2.062×10^7	4	2	2	Yes
G-SPSA ($\sigma^2 = 25, 5$)	2112	2.092×10^7	4	2	2	Yes

constraints.

Another comparison is done for several different initial guesses as shown in Table. 3.5. In this case study, we run the algorithm with several different initial guesses of the control variables as shown in Tables 3.5. Here, the control variables of initial guess 1 and initial guess 2 start from the middle of control variable range. The water injection rate controls of initial guess 3, initial guess 4, initial guess 5 and initial guess 6 start from the middle of control variable range. The BHP controls of initial guess 3 and initial guess 4 start close to the lower bound. Meanwhile the BHP controls of initial guess 5 and initial guess 6 start close to the upper bound. Note that in this comparison, we keep the same initial penalty parameter, $\mu_0 = 10^{-7}$,

Table 3.4: The performance of different algorithms obtained from one optimization with one initial random seed when we change initial penalty parameter (N_μ: Number of μ updated; N_λ: Number of λ updated; $M = 5$, $k_{\max} = 450$, $A = 45$, $a_0 = 1$, and $c_0 = 0.1$; $N_s = 2$ and $\sigma^2 = (1, 1)$; initial guess 1)

μ^0	$\lambda^0_{c,j}$	Simu-lations	Final NPV,$	Outer -loop	N_μ	N_λ	Conver-gence
10^{-1}	$\max[0, \frac{s_{c,j}c^0_j}{\mu^0}]$	2700	2.023×10^7	10	9	1	No
10^{-2}	$\max[0, \frac{s_{c,j}c^0_j}{\mu^0}]$	2670	2.018×10^7	10	9	1	Yes
10^{-3}	$\max[0, \frac{s_{c,j}c^0_j}{\mu^0}]$	2700	2.077×10^7	7	6	1	No
10^{-4}	$\max[0, \frac{s_{c,j}c^0_j}{\mu^0}]$	2424	2.021×10^7	8	7	1	Yes
10^{-5}	$\max[0, \frac{s_{c,j}c^0_j}{\mu^0}]$	2502	1.996×10^7	6	4	2	Yes
10^{-6}	$\max[0, \frac{s_{c,j}c^0_j}{\mu^0}]$	2268	1.922×10^7	5	4	1	Yes
10^{-7}	$\max[0, \frac{s_{c,j}c^0_j}{\mu^0}]$	2184	2.062×10^7	4	2	2	Yes
10^{-8}	$\max[0, \frac{s_{c,j}c^0_j}{\mu^0}]$	2346	2.063×10^7	2	1	1	Yes

and the initial Lagrange multipliers are estimated by Eq. 3.23. Control variables are sampled from Gaussian distribution with a pair variances of (1,1), of which the first number corresponds to the transformed variable of BHP control and the second number corresponds to the transformed variable of water injection rate control, and a correlation length of $N_s = 2$ (Eq. 2.15). Table. 3.6 present the performance of the G-SPSA-based augmented Lagrangian function using different initial guesses. In Table. 3.6, both initial guess 1 and initial guess 2 give a similar final NPV of around \$$2.062 \times 10^7$ (Table. 3.6). Although the number of iterations required to obtain convergence with initial guess 2 are far fewer. Here the results for initial guess 1 are still inferior to the results for initial guess 2. Because the water injection rate controls for initial guess 2 at each control step are equal to 65 STB/D, the initial FWIR's at each control step are greater than 2700 STB/D, so the constraint is violated, whereas with initial guess 1 of 60 STB/D, FWIR$\leq$ 2700 STB/D is satisfied at the initial guess 1. Thus with initial guess 2, the initial Lagrange multipliers stat at

Table 3.5: Several different initial guesses of control variables.

Initial Guess	Producer BPH control -	Water injection rate control
Initial Guess 1	5750 (psi)	60 STB/D
Initial Guess 2	5750 (psi)	65 STB/D
Initial Guess 3	3501 (psi)	60 STB/D
Initial Guess 4	3501 (psi)	65 STB/D
Initial Guess 5	5999 (psi)	60 STB/D
Initial Guess 6	5999 (psi)	65 STB/D

a non-zero value, whereas they are equal to zero with initial guess 1. It may be that the augmented Lagrangian function is less likely to suffer from ill condition. However, as these result are based on only one initial random seed, the result are far from conclusions. Perhaps the most important observation obtained from the result of Table. 3.6 is that we obtained significantly inferior NPV's with initial guesses 3 through 6. For all of this cases the initial guesses are essentially at the upper or lower bound so in the log-transform space the value of s is close to $\pm\infty$. For these values a perturbation in s has essentially no effect on NPV, i.e., the sensitivity of NPV to s is essentially zero so it is very difficult to change the value of the control variables during optimization. Moreover, with the initial stepsize order of 1, it take many iterations to move away from a bound. The most important lesson in this example is that the initial guess should not be close to a bound.

Finally, the results of this example suggest that the G-SPSA-based augmented Lagrangian algorithm can be applied effectively to maximize NPV subject to linear inequality constraints and bound constraints.

3.4.3 Production optimization with nonlinear and linear constraints

In this example, we implement production optimization for a model of horizontal reservoir which has a uniform grid system, $25 \times 25 \times 1$ with $\Delta x = \Delta y = 200$ ft. The thickness of the reservoir is 20 ft. The fluid system is incompressible two-phase

Table 3.6: The performance of different algorithms obtained from one optimization with one initial random seed when change initial guess (N_μ: Number of μ updated; N_λ: Number of λ updated; $M = 5$, $k_{\max} = 450$, $A = 45$, $a_0 = 1$, and $c_0 = 0.1$; $N_s = 2$ and $\sigma^2 = (1, 1)$).

Algorithm	Simu-lations	Initial NPV,$	Final NPV,$	Outer -loop	N_μ	N_λ	Conver-gence
Initial Guess 1	2184	1.246×10^7	2.062×10^7	6	4	2	Yes
Initial Guess 2	1248	1.264×10^7	2.066×10^7	5	3	2	Yes
Initial Guess 3	204	1.278×10^7	1.320×10^7	2	1	1	Yes
Initial Guess 4	912	1.295×10^7	1.501×10^7	5	3	2	Yes
Initial Guess 5	402	1.233×10^7	1.867×10^7	3	2	1	Yes
Initial Guess 6	324	1251×10^7	1.832×10^7	2	1	1	Yes

flow of water and oil. The porosity is homogeneous for the whole reservoir. The log-permeability distribution is shown in Fig. 3.11 and exhibits high permeability channels. The initial reservoir pressure is 3500 psi and the initial oil saturation is 0.2.

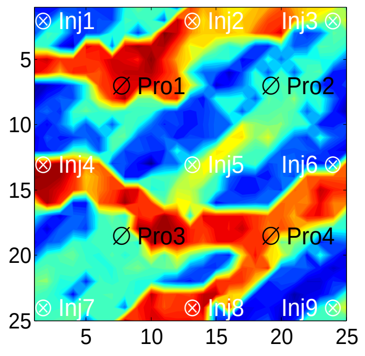


Figure 3.11: ln(k) distribution.

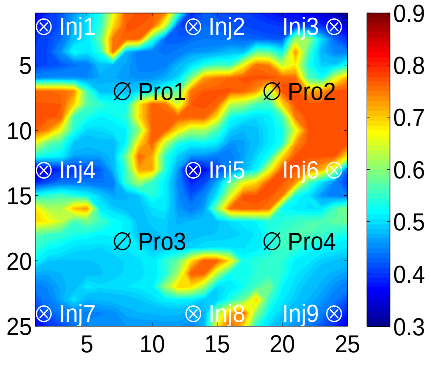


Figure 3.12: Final oil saturation with reactive control.

There are a total of 13 vertical wells with 4 production wells and 9 injection wells arranged in a five-spot well pattern, as shown in Fig. 3.11. The anticipated total project life is 1800 days and the control step size is equal to 60 days so we have 30 control steps. The total number of control variables is $(4 + 9) \times 30 = 390$. To optimize the NPV, we set each injection well under rate control with a lower bound of 0 STB/D and an upper bound of 2000 STB/D. The producer wells are under bottom hole pressure (BHP) control with a lower bound of 1500 psi and an upper bound of 6000 psi. The field water cut (FWCT) at each control step is required to be less than or equal to 0.7, i.e., the nonlinear constraint is FWCT $-$ 0.7 $\leq$ 0. The field water injection rate at each control step is required to be less than or equal to 5000 STB/D, i.e., the linear constraint is FWIR $-$ 5000 (STB/D) $\leq$ 0. All these inequality constraints are combined with the NPV to make up the augmented Lagrangian function. In this example, there is no equality constraint so the augmented Lagrangian function can be written as

$$\begin{aligned}\beta(u,y,\mu,\lambda) = J(u,y) &- \sum_{j=1}^{60} \lambda_{\mathrm{c},j}\left[\max\left\{c_j(u,y), -\lambda_{\mathrm{c},j}\frac{\mu}{s_{\mathrm{c},j}}\right\}\right] \\ &- \frac{1}{2\mu}\sum_{j=1}^{60} s_{\mathrm{c},j}\left[\max\left\{c_j(u,y), -\lambda_{\mathrm{c},j}\frac{\mu}{s_{\mathrm{c},j}}\right\}\right]^2.\end{aligned} \tag{3.32}$$

where thirty non-linear inequality constraints for FWCT are calculated by

$$c_j = \mathrm{FWCT}^h - 0.7 \leq 0;\ h = 1, \ldots, 30 \text{ and } j = h, \tag{3.33}$$

and thirty linear inequality constraints for FWIR are calculated by

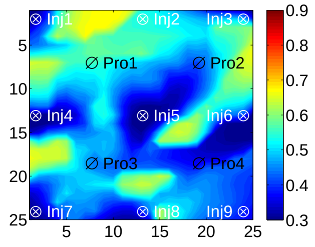
$$c_j = \sum_{i=1}^{9} q_{winj,i}^h - 5000\ (\mathrm{STB/D}) \leq 0;\ h = 1, \ldots, 30, \text{ and } j = h + 30, \tag{3.34}$$

where i denotes the injection well index, h denotes the control step index, and j denotes the constraint index. The oil price is set at \$50/STB, the water injection rate cost at \$0/BBL, the water production cost at \$5.56/STB, and the annual discount rate is 10%.

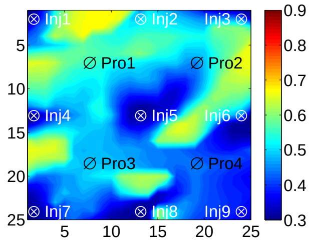
Here, the average of five SPSA gradients (Eq. 2.17) calculated by one-sided simultaneous perturbation is applied to generate a steepest-ascent direction. We use the normalized search direction (Eq. 2.21) to update control vector. After working with both the B-SPSA gradient and the G-SPSA gradient in some previous examples, we see that the results coming from the B-SPSA gradient is always worse than those coming from the G-SPSA gradient. Therefore, from this example, we only consider the G-SPSA-based augmented Lagrangian algorithm where control variables are sampled from Gaussian distribution with a variances pair of (1,1), of which the first number corresponds to the transformed variable of BHP control and the second number corresponds to the transformed variable of water injection rate control, and a correlation length of $N_s = 30$ (Eq. 2.15). The value of α and γ in those equations is again $\alpha = 0.602$ and $\gamma = 0.101$. By setting the maximum number of allowable iterations equal to 390 which is identical to the number of control variables, then we obtain the value of A equal to 39. By setting $a_0 = 1.5$, we obtain $a = 14.03$. We set $c_{\min} = 0.547$ which gives $c_0 = 0.1$.

The initial guess for control variables is 300 STB/D for each injection well and 3500 psi for each production well at all control steps (initial guess 1). We set the scaling factors for inequality constraints $s_{c_j} = 1/C_j$ for all j's. For field water injection rate constraints, C_j is set to 0.7, $j = 1, \ldots, 30$, and for field water injection rate constraints, C_j is set to 5000 STB/D, $j = 31, \ldots, 60$. The initial penalty parameter is set to 10^{-7}. As discussed previously, we suspect that ideally this 10^7 value should be on the order of 1/10 the expected value of J. The initial Lagrange multipliers are estimated by using Eq. 3.23. The initial values of the convergence tolerances in Eqs. 3.28 and 3.29 are $\Delta\beta_{k,\ell} \le \epsilon_0 = 0.001$, $\Delta u_{k,\ell} \le \xi_0 = 0.05$, respectively, and these values decrease very slowly from outer-loop iteration to outer-loop iteration. The initial tolerance of the violation of the constraints is set to 0.0025, i.e., $\eta^0 = 0.0025$. The algorithm terminates when $\Delta\beta_{k,\ell} \le \epsilon^* = 0.6\epsilon_0$, $\Delta u_{k,\ell} \le \xi^* = 0.6\xi_0$ and $\sigma_{c_\nu} \le \eta^* = 0.01\eta_0$.

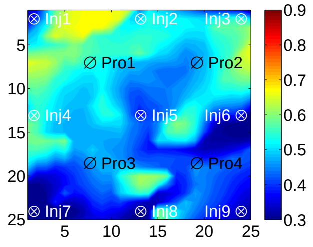
Fig. 3.12 shows the remaining oil saturation distribution in the reservoir obtained with reactive control and Fig. 3.13 shows the corresponding results obtained with optimization algorithms using initial guess 1. In this case, reactive control refers to the procedure where we simply shut in a well whenever the cost of disposing of the produced water exceeds the oil revenue, i.e., when the WOR at the production well exceeds $50.0/5.56 = 9.0$. All optimization algorithms clearly result in a better sweep efficiency and oil recovery than are obtained by applying reactive control.



(a) G-SPSA

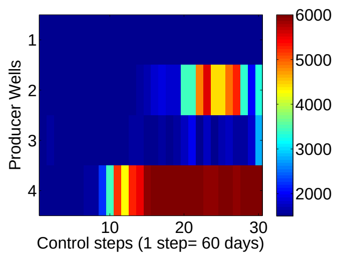


(b) ComG-SPSA1

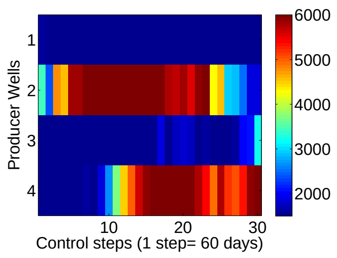


(c) ComG-SPSA2

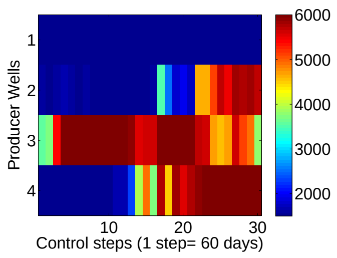
Figure 3.13: Remaining oil saturation distribution obtained from one optimization run with one initial random seed ($M = 5$, $k_{\max} = 390$, $A = 39$, $a_0 = 1.5$, and $c_0 = 0.1$; $N_s = 30$ and $\sigma^2 = (1, 1)$; $\mu_0 = 10^{-7}$; Initial guess 1).



(a) G-SPSA



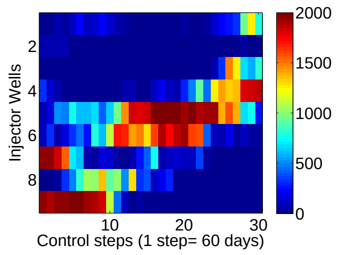
(b) ComG-SPSA1



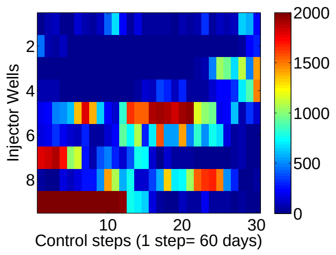
(c) ComG-SPSA2

Figure 3.14: The estimated optimal BHP well controls obtained from one optimization run with one initial random seed ($M = 5$, $k_{\max} = 390$, $A = 39$, $a_0 = 1.5$, and $c_0 = 0.1$; $N_s = 30$ and $\sigma^2 = (1, 1)$; $\mu_0 = 10^{-7}$; Initial guess 1).

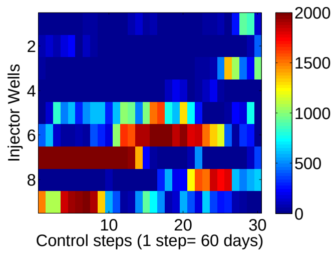
The final well controls for the production wells and injection wells obtained by different algorithms using initial guess 1 are shown in Figs. 3.14 and 3.15. The y-axis in each sub-figure of Fig 3.14 corresponds to the four BHP controls. The x-axis in each sub-figure corresponds to the thirty control steps. The color scale



(a) G-SPSA



(b) ComG-SPSA1



(c) ComG-SPSA2

Figure 3.15: The estimated optimal water injection rate well controls obtained from one optimization run with one initial random seed ($M = 5$, $k_{\max} = 390$, $A = 39$, $a_0 = 1.5$, and $c_0 = 0.1$; $N_s = 30$ and $\sigma^2 = (1, 1)$; $\mu_0 = 10^{-7}$; Initial guess 1).

corresponds to bottom- hole pressure of the production wells, with blue representing the lowest bottom-hole pressure of producer well (1500 psi) and red representing highest allowable bottom-hole pressure at which the production well is shut-in. The y-axis in each sub-figure of Fig 3.15 corresponds to the nine water injection controls, and the x-axis corresponds to the thirty control steps. The color scale corresponds to the injection rates, with white representing a rate essentially equal to close to zero and red representing the highest rate.

For the G-SPSA and ComG-SPSA1 algorithms, in the area around the bottom high permeability channel, the production well Pro4 is shut in at some of the later control steps as the water approaches production well Pro4 (see Figs. 3.13 and 3.14). Injection well Inj9 injects at the highest rate at early control steps (see see Figs. 3.13 and 3.15). When the water approaches production well Pro4, the injection rate of injection well Inj9 reduces to a lower value. Injection well Inj7 also injects with a high rate (around 2000 STB/D) at some early control steps. The injection well Inj8 injects with a rate about 1200 STB/D at some middle control steps and the injection well Inj5 and Inj6 still inject with a high injection rate around 2000 STB/D in the middle control steps. Overall this strategy minimizes the water approaching production well Pro3, and production well Pro3 remains open for the whole life of the reservoir.

For the ComG-SPSA2 algorithm, the results are quite different as production well Pro3 is shut in at almost all control steps and the production well Pro4 is only shut in at some of the last control steps. Production well Pro3 is shut-in at most control steps is that injection well Inj7 and Inj9 operate at a high injection rate at early control steps. In the top high permeability channel, for all algorithms the production well Pro2 is shut in at some early control steps because Inj5 works with a high injection rate at middle control steps. BPH controls of the production well Pro1 always remain close to the lower bound of 2000 psi. Thus production well Pro1 produces throughout the whole life of reservoir as there is still oil around this well and water does not breakthrough in the well. From these overall results, oil is primarily swept from bottom to top. It is consistent with the remaining oil saturation result of Fig. 3.13.

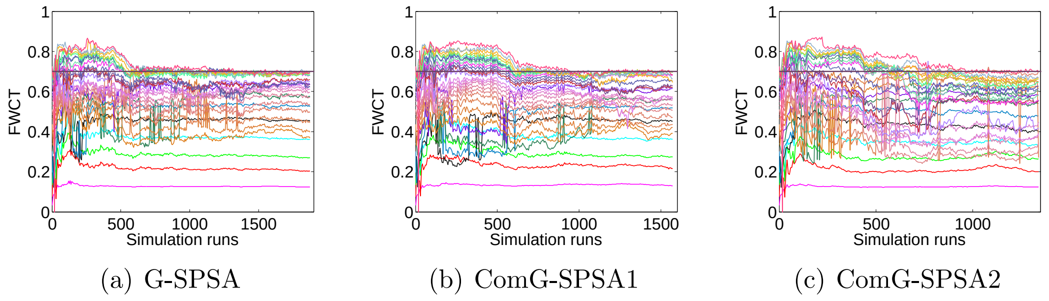
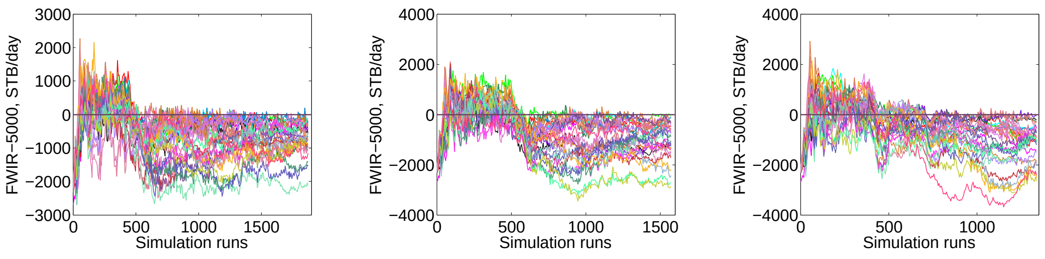


Figure 3.16: Constraint violation of FWCT versus simulation runs obtained from one optimization run with one initial random seed (($M = 5$, $k_{\max} = 390$, $A = 39$, $a_0 = 1.5$, and $c_0 = 0.1$; $N_s = 30$ and $\sigma^2 = (1, 1)$; $\mu_0 = 10^{-7}$; Initial guess 1)) .

Fig. 3.16 presents the relationship between the violation of the field water cut constraints and the number of simulation runs with initial guess 1. In Fig. 3.16, whenever FWCT is greater than 0.7, the constraint is violated. The thirty curves for FWCT correspond to the thirty control steps. Fig. 3.17 presents the relationship between the violation of the field water injection rate constraints and the number of

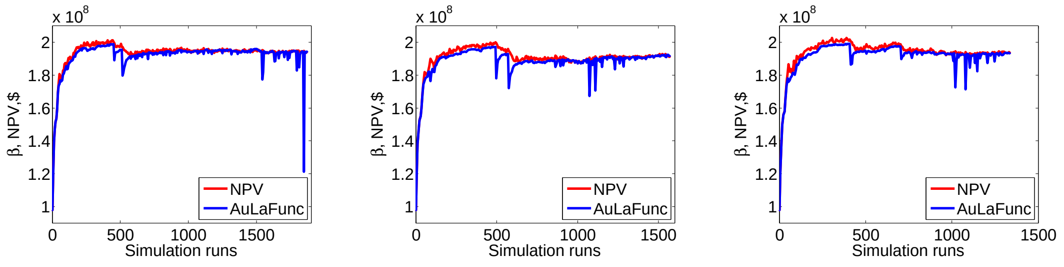


(a) G-SPSA (b) ComG-SPSA1 (c) ComG-SPSA2

Figure 3.17: Constraint violation of FWIR versus simulation runs obtained from one optimization run with one initial random seed (($M = 5$, $k_{\max} = 390$, $A = 39$, $a_0 = 1.5$, and $c_0 = 0.1$; $N_s = 30$ and $\sigma^2 = (1,1)$; $\mu_0 = 10^{-7}$; Initial guess 1)).

simulation runs. We plot the field water injection rate (FWIR) minus 5000 STB/D versus the number of simulation runs, so whenever FWIR − 5000 (STB/D) > 0 the constraint is violated. The thirty curves for FWIR−5000 (STB/D) correspond to the thirty control steps. The algorithm terminates when $\Delta\beta_{k,\ell} \le 0.0006$, $\Delta u_{k,\ell} \le 0.03$, the violation of the constraints of FWCT is less than or equal to 0.0035 (0.5% of the constraint FWCT value 0.7), and the violation of FWIR constraints is less than or equal to 25 STB/D (0.5% of the constraint FWIR value 5000 STB/D. Note that with the initial guess 1, at some early simulation runs, many constraints are violated at early iterations but ultimately the violation of the constraints is satisfied to within the prescribed tolerance. Thus the results illustrate the G-SPSA-based augmented Lagrangian function works well not only with linear constraints but also with nonlinear constraints.

The NPV value and the augmented Lagrangian function value are plotted as a function of simulation runs in Fig. 3.18. The y-axis in each sub-figure of this figure corresponds to the NPV value and the augmented Lagrangian function value. The x-axis of these sub-figures corresponds to simulation runs. Note that significant decreases in the augmented Lagrangian function correspond to reducing the penalty parameter. Table 3.7 presents the performance of all G-SPSA-based augmented



(a) G-SPSA (b) ComG-SPSA1 (c) ComG-SPSA2

Figure 3.18: NPV versus simulation runs obtained from one optimization run with one initial random seed ($M = 5$, $k_{\max} = 390$, $A = 39$, $a_0 = 1.5$, and $c_0 = 0.1$; $N_s = 30$ and $\sigma^2 = (1, 1)$; $\mu_0 = 10^{-7}$; Initial guess 1).

Table 3.7: The performance of different algorithms obtained from one optimization with one initial random seed (N_μ: Number of μ updated; N_λ: Number of λ updated; $M = 5$, $k_{\max} = 390$, $A = 39$, $a_0 = 1.5$, and $c_0 = 0.1$; $N_s = 30$ and $\sigma^2 = (1, 1)$; $\mu_0 = 10^{-7}$; Initial guess 1).

Algorithm	Simu-lations	Final NPV,$	Outer -loop	N_μ	N_λ	Conver-gence
G-SPSA ($N_s = 30$)	1872	1.941×10^8	7	5	2	Yes
ComG-SPSA1 ($N_s = 30$)	1572	1.911×10^8	5	3	2	Yes
ComG-SPSA2 ($N_s = 30$)	1338	1.934×10^8	4	3	1	Yes

Lagrangian algorithms. All three SPSA algorithms based on Gaussian perturbation converged to similar estimates of optimal NPV's although ComG-SPSA2 required far fewer simulation runs to obtain convergence.

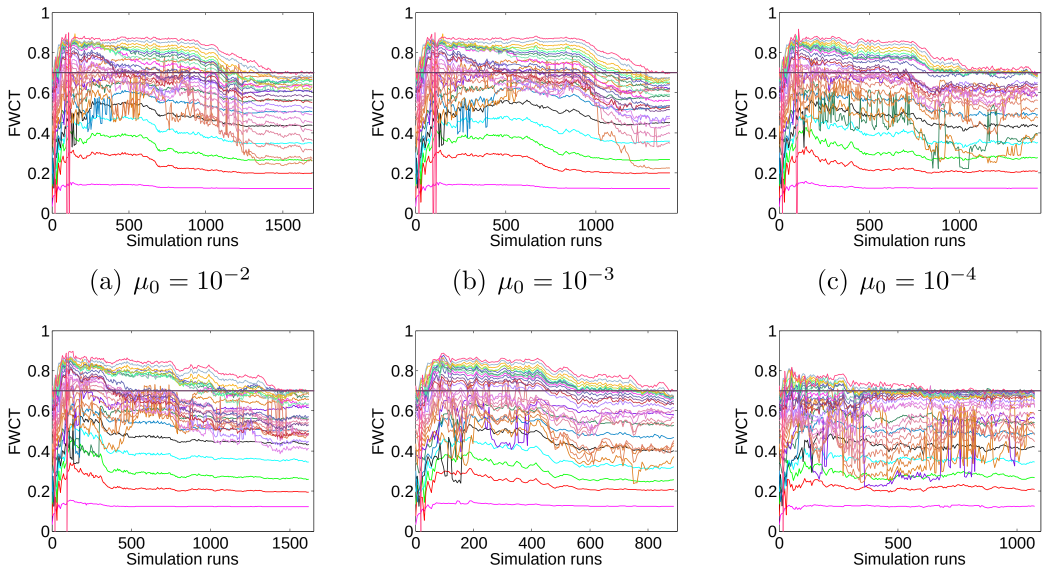
Table 3.8 compares G-SPSA algorithm's performance variances of (1,1) and (25,5) using initial guess 1. The first number in each pair of variance corresponds to the transformed variables of BHP control and the second number corresponds to the transformed variables of water injection rate control. The initial penalty parameter is set to 10^{-7} and the initial Lagrange multipliers are calculated by Eq. 3.23. We see that for both pairs of variances, the G-SPSA-based augmented Lagrangian algorithm converges. The variance pair of (1,1) gives a slightly higher NPV value than is obtained with (25,5) variances but requires twice as many iteration to require convergence.

Table 3.8: The performance of different algorithms obtained from one optimization with one initial random seed when we change the value of variances (N_μ: Number of μ updated; N_λ: Number of λ updated; $M = 5$, $k_{\max} = 390$, $A = 39$, $a_0 = 1.5$, and $c_0 = 0.1$; $N_s = 30$ and $\sigma^2 = (1, 1)$; $\mu_0 = 10^{-7}$; Initial guess 1).

Algorithm	Simu-lations	Final NPV,$	Outer -loop	N_μ	N_λ	Conver-gence
G-SPSA ($\sigma^2 = 1, 1$)	1872	1.941×10^8	7	5	2	Yes
G-SPSA ($\sigma^2 = 25, 5$)	780	1.929×10^8	5	4	1	Yes

The violation of FWCT and FWIR constraints with different initial penalty parameters are respectively presented in Fig. 3.19 and 3.20. Again, the y-axis in each sub-figure of these figures corresponds to the value of the constraint violation of FWCT and FWIR, and the x-axis in these sub-figures corresponds to the number of simulation runs. The 30 curves in each sub-figure of these figures corresponds respectively to FWCT − 0.7 and FWIR − 5000 STB/D. We can see that when the algorithm terminates the violation of FWCT and FWIR constraints are always less than the final tolerance of constraint violation. For every initial penalty parameter used in this example, the constraints are satisfied to within the given tolerance and convergence is obtained. Table. 3.9 presents the performance of the G-SPSA algorithm using several different initial penalty parameters, the final NPV's are similar in all cases but the $\mu_0 = 10^{-6}$ and $\mu_0 = 10^{-7}$ case are somewhat abnormal as they require far fewer iterations to convergence. However, this results is not general and the process is stochastic we cannot draw any hand conclusions from these results.

In this example, we also test the algorithm with several different initial guesses as shown in Table. 3.10. Initial guess 1 and initial guess 2 start from the middle of control variable range. However, the initial Lagrange multipliers corresponding to FWIR constraints at the initial guess 1 is set to zero as they are estimated by Eq. 3.23. Meanwhile, the initial Lagrange multipliers corresponding to FWIR constraints at



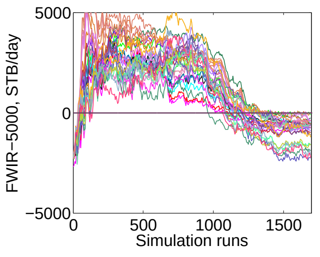
(a) $\mu_0 = 10^{-2}$ (b) $\mu_0 = 10^{-3}$ (c) $\mu_0 = 10^{-4}$

(d) $\mu_0 = 10^{-5}$ (e) $\mu_0 = 10^{-6}$ (f) $\mu_0 = 10^{-8}$

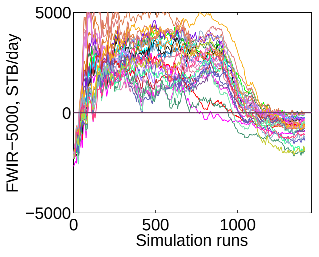
Figure 3.19: Constraint violation of FWCT versus simulation runs obtained from one optimization run with one initial random seed when we change the initial penalty parameter ($M = 5$, $k_{\max} = 390$, $A = 39$, $a_0 = 1.5$, and $c_0 = 0.1$; $N_s = 30$ and $\sigma^2 = (1, 1)$; Initial guess 1).

the initial guess 2 is not equal to zero. The injection rate controls of the other initial guesses also start from the middle of control variable range. Initial guess 3 and initial guess 4 have the BHP control variables are approximately equal to the lower bound of 1501 psi. The BHP control variables of initial guess 5 and initial guess 6 are approximately equal to the upper bound of 5999 psi. Note that this comparison is done while we keep the same initial penalty parameter, $\mu_0 = 10^{-7}$, and the initial Lagrange multipliers are estimated by Eq. 3.23. Control variables are sampled from Gaussian distribution with a variance pair of (1,1), of which respectively corresponds to the transformed variables of BPH control and the transformed variables of water injection control, and a correlation range of $N_s = 30$.

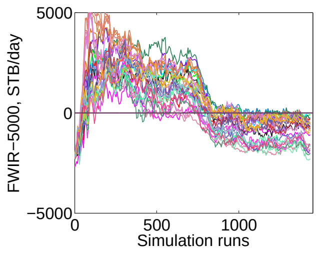
Fig. 3.21 presents the final controls for injection wells as the water injection rates for each initial guess are from the middle of the allowable range. There are significant difference in the final BPH control variables between initial guesses, es-



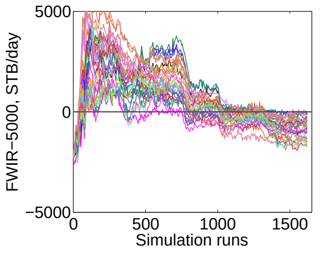
(a) $\mu_0 = 10^{-2}$



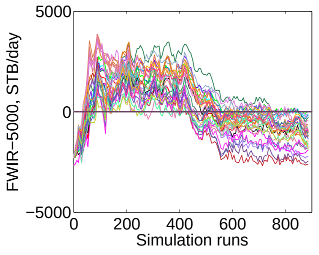
(b) $\mu_0 = 10^{-3}$



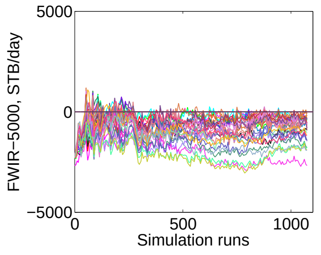
(c) $\mu_0 = 10^{-4}$



(d) $\mu_0 = 10^{-5}$



(e) $\mu_0 = 10^{-6}$



(f) $\mu_0 = 10^{-8}$

Figure 3.20: Constraint violation of FWIR versus simulation runs obtained from one optimization run with one initial random seed when we change the initial penalty parameter ($M = 5$, $k_{\max} = 390$, $A = 39$, $a_0 = 1.5$, and $c_0 = 0.1$; $N_s = 30$ and $\sigma^2 = (1, 1)$; Initial guess 1).

pecially from initial guess 3 to initial guess 6, which have the initial guess of BPH control variables approximately equal to the upper bound or the lower bound (see Fig. 3.22).

The performance of the G-SPSA-based augmented Lagrangian algorithm using different initial guesses is shown in Table 3.11. Similar to the example in Subsection 3.4.2, initial guess 1 and initial guess 2 yield by far the highest NPV's about $\$1.94 \times 10^8$. However, with initial guess 2, the algorithm converges after 1248 simulation runs, whereas with initial guess 1, 1823 reservoir simulation runs are required to obtain convergence. Because the water injection rate controls of initial guess 2 at each control step are set to 560 STB/D, the initial FWIR's at each control step are greater than constraint of 5000 STB/D so the initial Lagrange multipliers are non-zero. As in the example of subsection 3.4.2, starting from non-zero values of λ's seems to speed convergence possibility because it alleviates the effect of ill condition.

Table 3.9: The performance of different algorithms obtained from one optimization with one initial random seed when we change the initial penalty parameters (N_μ: Number of μ updated; N_λ: Number of λ updated; $M = 5$, $k_{\max} = 390$, $A = 39$, $a_0 = 1.5$, and $c_0 = 0.1$; $N_s = 30$ and $\sigma^2 = (1, 1)$; $\mu_0 = 10^{-7}$; Initial guess 1).

μ^0	$\lambda^0_{c,j}$	Simu-lations	Final NPV,\$	Outer -loop	N_μ	N_λ	Conver-gence
10^{-1}	$\max[0, \frac{s_{c,j}c^0_j}{\mu^0}]$	1614	1.930×10^8	10	9	1	Yes
10^{-2}	$\max[0, \frac{s_{c,j}c^0_j}{\mu^0}]$	1692	1.976×10^8	9	8	1	Yes
10^{-3}	$\max[0, \frac{s_{c,j}c^0_j}{\mu^0}]$	1410	1.959×10^8	8	7	1	Yes
10^{-4}	$\max[0, \frac{s_{c,j}c^0_j}{\mu^0}]$	1434	1.932×10^8	8	7	1	Yes
10^{-5}	$\max[0, \frac{s_{c,j}c^0_j}{\mu^0}]$	1620	1.950×10^8	6	6	0	Yes
10^{-6}	$\max[0, \frac{s_{c,j}c^0_j}{\mu^0}]$	888	1.910×10^8	4	4	0	Yes
10^{-7}	$\max[0, \frac{s_{c,j}c^0_j}{\mu^0}]$	1872	1.941×10^8	7	5	2	Yes
10^{-8}	$\max[0, \frac{s_{c,j}c^0_j}{\mu^0}]$	1074	1.923×10^8	3	1	2	Yes

Note that with the value of the water injection rate controls of initial guess 1, the initial Lagrange multipliers are equal to zero. Also as in the example of subsection 3.4.2, starting with initial guesses close to their bound is a poor choice even though the some optimal well controls may be equal to their bounds (see Fig. 3.22).

Finally, as other examples of this chapter, this example illustrates that the G-SPSA-based augmented Lagrangian algorithm can work well to maximize the NPV problems subject to linear and non-linear inequality constraints. Also the initial guesses for controls should not be set equal to the bounds for the controls. Initial guesses where the BPH control variables are close to the upper bound or the lower bound are also not a good choice. The results again suggest that for constrained optimization problems, we should choose initial guesses that can make the initial Lagrange multipliers calculated by Eq. 3.23 non-zero as this condition enhances the performance of the algorithm. A good value of the initial penalty parameter is to set $1/\mu_0$ equal to 10% of the expected value of NPV.

Table 3.10: Several different initial guess of control variables.

Initial Guess	Producer BPH control -	Water injection rate control
Initial Guess 1	3500 (psi)	300 STB/D
Initial Guess 2	3500 (psi)	560 STB/D
Initial Guess 3	1501 (psi)	300 STB/D
Initial Guess 4	1501 (psi)	560 STB/D
Initial Guess 5	5999 (psi)	300 STB/D
Initial Guess 6	5999 (psi)	560 STB/D

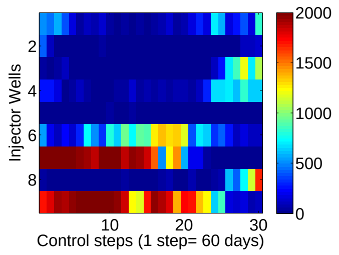
Although the immediately preceding conclusion is correct for this example, it is not a valid conclusion for other examples. Moreover, to make a consistent conclusion for this example, we also should run the G-SPSA-based augmented Lagrangian algorithm with several different initial random seeds. The computational average results might give a reasonable conclusion that is applicable for other examples as well.

Table 3.11: The performance of different algorithms obtained from one optimization with one initial random seed when we change initial guess (N_μ: Number of μ updated; N_λ: Number of λ updated; $M = 5$, $k_{\max} = 390$, $A = 39$, $a_0 = 1.5$, and $c_0 = 0.1$; $N_s = 30$ and $\sigma^2 = (1, 1)$; $\mu_0 = 10^{-7}$).

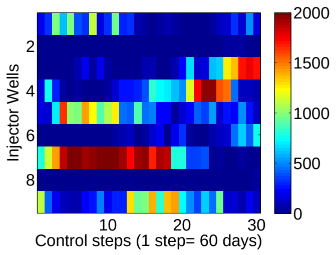
Algorithm	Simu-lations	Initial NPV,$	Final NPV,$	Outer -loop	N_μ	N_λ	Conver-gence
Initial Guess 1	1872	0.975×10^8	1.941×10^8	6	4	2	Yes
Initial Guess 2	1248	1.176×10^8	1.954×10^8	5	3	2	Yes
Initial Guess 3	2340	1.656×10^8	1.833×10^8	4	3	1	No
Initial Guess 4	2340	1.797×10^8	1.863×10^8	3	2	1	No
Initial Guess 5	1638	0.234×10^8	0.611×10^8	5	4	1	Yes
Initial Guess 6	1686	0.476×10^8	0.632×10^9	5	3	2	Yes

3.4.4 Production optimization for Brugge case with inequality and simple bound constraints

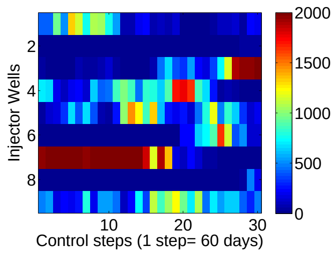
In this example, we use the same reservoir model and well data as in the



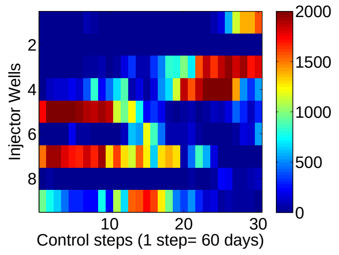
(a) Initial Guess 2



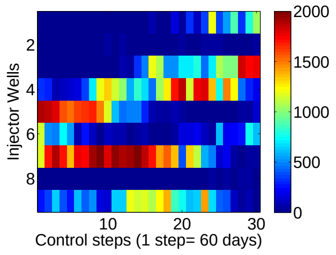
(b) Initial Guess 3



(c) Initial Guess 4



(d) Initial Guess 5

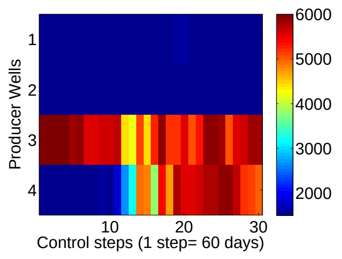


(e) Initial Guess 6

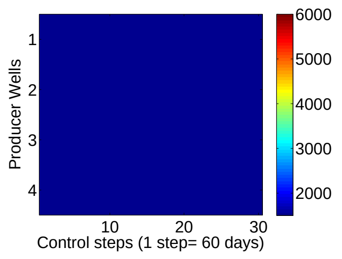
Figure 3.21: The estimated optimal water injection rate well controls obtained from one optimization run with one initial random seed when we change initial guess ($M = 5$, $k_{\max} = 390$, $A = 39$, $a_0 = 1.5$, and $c_0 = 0.1$; $N_s = 30$ and $\sigma^2 = (1, 1)$) .

Brugge case in Chapter 1. Recall that there are 30 vertical wells in the Brugge reservoir, including 20 smart producing wells and 10 smart water injection wells. Each well has three segments that can be controlled individually by inflow control valves. We optimize the well controls for years 10 through 30 based on the mean model obtained by Chen et al. (2010) using the ensemble Kalman filter with covariance localization to assimilate production data for the first ten years of the reservoir life. This example is identical to the production optimization problem for years 10-30 solved with an adjoint-gradient method in Chen et al. (2010).

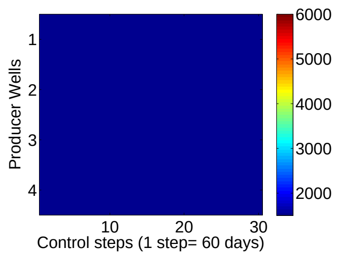
The production period of the reservoir is divided into 40 control steps, i.e., each control step is 182.5 days. There are 84 control variables for each control step. These control variables are the liquid production rate at each individual segment of the production wells and the water injection rate at each individual segment of the injection wells. The total number of control variables is $40 \times 84 = 3360$. The



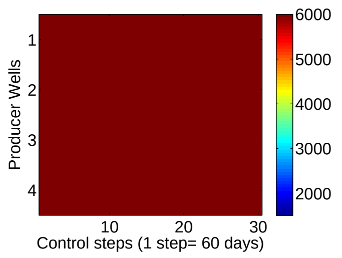
(a) Initial Guess 2



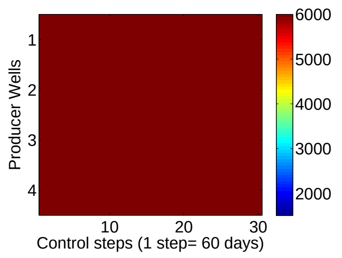
(b) Initial Guess 3



(c) Initial Guess 4



(d) Initial Guess 5



(e) Initial Guess 6

Figure 3.22: The estimated optimal BHP well controls obtained from one optimization run with one initial random seed when we change initial guess ($M = 5$, $k_{\max} = 390$, $A = 39$, $a_0 = 1.5$, and $c_0 = 0.1$; $N_s = 30$ and $\sigma^2 = (1, 1)$)

maximum liquid production rate of each producer segment is 3000 STB/D, and the maximum injection rate of each injector segment is 4000 STB/D. The minimum value for the rate of each segment in a production or an injection well is 0 STB/D. The minimum BHP constraint for a producer segment is 725 psi; the maximum BHP constraint for an injection well segment is 2611 psi. The BHP nonlinear constraints are considered reactively by inputting them directly into the simulator data file. In addition to the bound constraints, we have a large number of inequality constraints. The total liquid production rate of the three segments of each production well j must be less than or equal to 3000 STB/D, i.e.,

$$q_{L_j,1} + q_{L_j,2} + q_{L_j,3} \leq 3000 \text{ (STB/D)}, \; j = 1, 2, \ldots, 20. \tag{3.35}$$

Similarly the water injection rates at three segments of each injector must satisfy

$$q_{\mathrm{inj}_j,1} + q_{\mathrm{inj}_j,2} + q_{\mathrm{inj}_j,3} \le 4000\ (\mathrm{STB/D}),\ j = 1, 2, \ldots, 10. \tag{3.36}$$

These 1200 linear inequality constraints are incorporated into the augmented Lagrangian function of Eq. 3.10. The oil price is $r_{\mathrm{o}} = \$80.0/\mathrm{STB}$, and both the water production and the injection costs are $r_{\mathrm{w}} = r_{\mathrm{winj}} = \$5.0/\mathrm{STB}$. The annual discount rate is 10%.

The initial value for the injection rate of each injection well segment is 1333.0 STB/D, and the initial value for the liquid production rate of each production well segment is 700 STB/D. We set the scaling factors for inequality constraints $s_{\mathrm{c}_i} = 1/C_i^2$. For the total liquid production rate constraints of the producers, $C_i = 3000$ STB/D, $i = 1, 2, \cdots, 800$ and for the total injection rate of the injectors, $C_i = 4000$ STB/D, $i = 801, 802, \cdots, 1200$. The initial Lagrange multipliers are calculated by Eq. 3.23 and the initial penalty parameter is set equal to 10^{-7}. The initial values of the convergence tolerances in Eqs. 3.28 and 3.29, respectively, are set to $\Delta\beta_{k,\ell} \le \epsilon_0 = 0.005$ and $\Delta u_{k,\ell} \le \xi_0 = 0.05$, and these values decrease very slowly from outer-loop iteration to outer-loop iteration. The initial tolerance of the violation of the constraints is equal to 0.0025, i.e., $\eta^0 = 0.0025$. The algorithm terminates when $\Delta\beta_{k,\ell} \le \epsilon^* = 0.4\epsilon_0 = 0.002$, $\Delta u_{k,\ell} \le \xi^* = 0.4\xi_0 = 0.02$ and $\sigma_{c_\nu} \le \eta^* = 0.01\eta_0$, i.e., the violation of the constraints is less than or equal to 30 STB/D. The search direction in Eq. 2.18 is computed from Eq. 2.17 with $M = 10$. A variance pair of (1,1), of which the first number corresponds to the transformed variable of liquid production rate and the second number corresponds to the transformed variable of water injection rate, and $N_s = 40$ are applied to generate the covariance matrix C_U (Eq. 2.15) which we use to generate perturbations for calculation of stochastic gradients and for promoting temporal smoothness of the well controls at each individual well. We set $\alpha = 0.602$ and $\gamma = 0.101$ which are the same as recommendation of Spall (1998). The maximum number of iterations allowable is set to the number of controls, i.e., set equal to 3360, we obtain $A = 336$. By setting the initial stepsize equal to 1.5, we obtain $a = 50.0$. By setting $c_{\min} = 0.05$, we obtain $c = 0.113$.

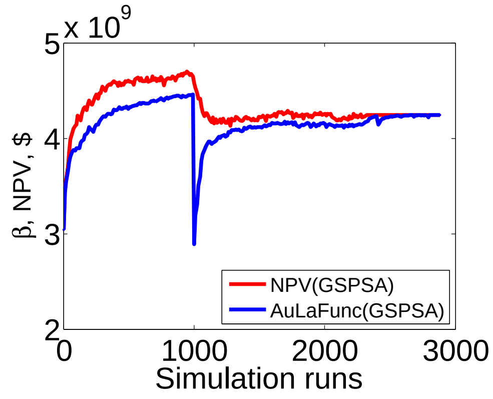


Figure 3.23: Objective functions versus simulation runs ($\mu_0 = 10^{-7}$).

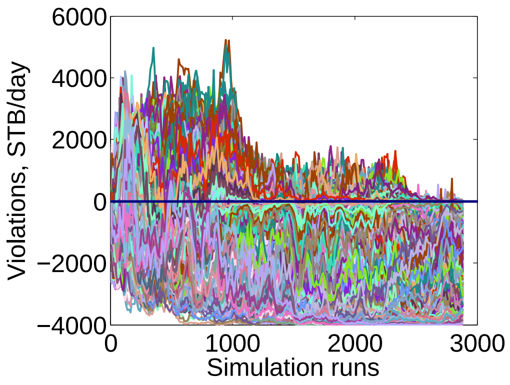
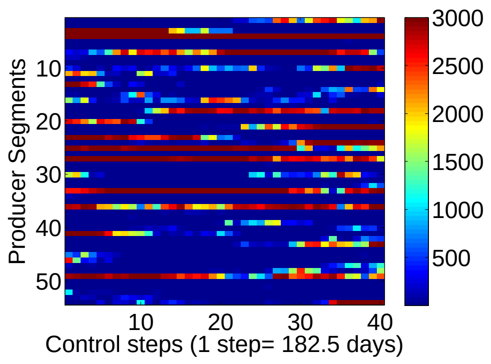
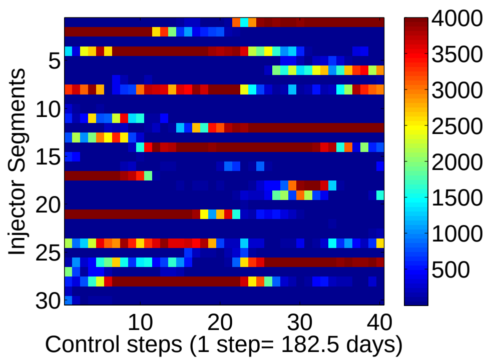


Figure 3.24: Constraint violation versus simulation runs ($\mu_0 = 10^{-7}$).



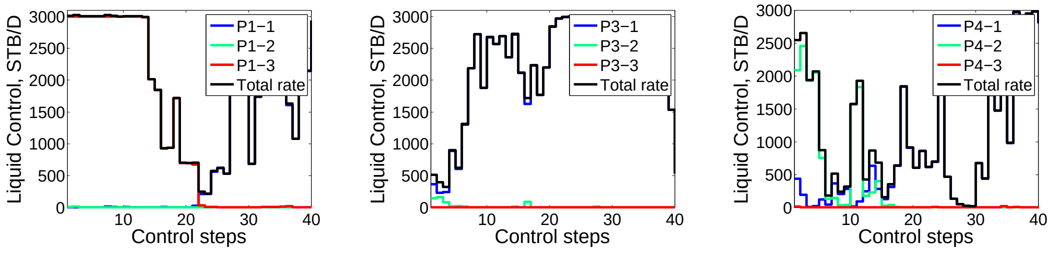
(a)



(b)

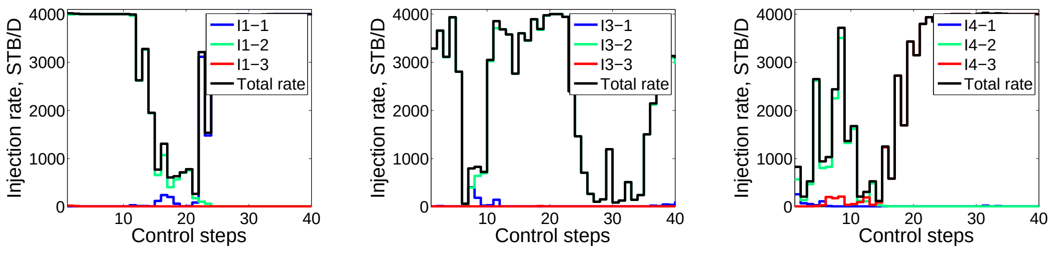
Figure 3.25: The estimated optimal well segment liquid production rates

The G-SPSA-based augmented Lagrangian algorithm converged after 2882 simulation runs using three outer loop iterations. The NPV increased from $\$3.04 \times 10^9$ at the initial guess for optimal well controls to $\$4.25 \times 10^9$ at convergence. Using an augmented Lagrangian approach with gradients computed by the adjoint method, Chen et al. (2010) achieved a realized NPV of $\$4.17 \times 10^9$, which is about 2% lower than our final NPV value. The Chen et al. result, however, only required 30 reservoir simulation runs, two orders of magnitude fewer simulation runs than we required to achieve our result. It is important to realize, however, the preconditioned steepest ascent algorithm does not require adjoint capability, and to the best of our knowledge, no commercial reservoir simulator provides the user the gradient of general nonlinear constraints. The constraints for the Brugge case are linear. The G-SPSA-based augmented Lagrangian method presented here has been applied to problems with general nonlinear constraints as shown in previous example. The augmented Lagrangian function and NPV are plotted versus the number of reservoir simulation runs in Fig. 3.23, and the violation for all the group liquid production rate (GLPR) constraints and the group water injection rate (GWIR) constraints are shown in Fig. 3.24, where each individual curve corresponds to a specific well at a specific control step. At convergence, all the GLPR constraints and the GWIR constraints are satisfied within the small tolerance specified. The estimated optimal well controls at three production wells are shown in Fig. 3.26, corresponding results for the three injection wells are shown in Fig. 3.27. Note that at the end of the reservoir life, most injector well segments operate at or close to the lower bound of 0 STB/D.



(a) Pro 1 (b) Pro 3 (c) Pro 4

Figure 3.26: The estimated optimal well segment liquid production rates.



(a) Inj 1 (b) Inj 3 (c) Inj 4

Figure 3.27: The estimated optimal well segment injection rates.

CHAPTER 4

THEORETICAL COMPARISON OF SOME DERIVA- TIVE-FREE ALGORITHMS FOR PRODUCTION OPTIMIZATION

4.1 Ensemble-Based Optimization (EnOpt)

EnOpt (ensemble-based optimization) has its roots in the work of Nwaozo (2006) and Lorentzen et al. (2006), but the name EnOpt was coined by Chen et al. (2009). For bang-bang control problems (Sudaryanto and Yortsos, 2000; Zandvliet et al., 2006), Wang et al. (2009) found that using "EnOpt" to compute an approximate gradient of the NPV function did not produce the optimal bang-bang controls; however, in most cases, we expect and prefer that well controls vary smoothly in time. When the number of control variables is large, there may be several sets of controls that give the same optimal value of the NPV objective function (van Essen et al., 2009). In this case, the optimization problem has additional degrees of freedom which may be used to achieve a second objective, e.g., to optimize NPV over the short-term without compromising the long term (life cycle) NPV of production (van Essen et al., 2009; Chen et al., 2011b) or finding controls which are smooth in time but still maximize the NPV of production over the life of the reservoir. For optimal control problems which have multiple maxima, smoothly varying controls are desirable for field implementation and in this case, EnOpt, as implemented by Chen et al. (2009), appears to be a reasonably good choice. With the standard implementation of EnOpt for robust optimization, a realization of the vector of all well control variables is generated from a prescribed Gaussian $N(\bar{u}, C_U)$ for each reservoir model, and the associated value of NPV is calculated by running the reservoir simulator for the expected life of the reservoir. An approximation of the gradient preconditioned by multiplication by C_U is then approximated from the covariance between the control vector and NPV, where this covariance matrix is approximated from the ensemble of pairs of controls and NPV's. Even though we do not consider robust optimization, we can still generate multiple perturbations of the control vector about the current estimated optimum and use these perturbed control vectors with the corresponding values of NPV to estimate a preconditioned gradient.

Applying EnOpt to maximize the NPV as a function of the well controls for a given reservoir model requires the generation of an ensemble of controls. This is done by specifying a covariance matrix C_U for the control vector and generating an ensemble of N_e perturbed controls

$$\widehat{u}_j^\ell = u^\ell + C_U^{1/2} Z_j, \; j = 1, 2, \ldots, N_e, \tag{4.1}$$

where Z_j is an n_u-dimensional vector with its components given by independent standard random normal deviates and $C_U^{1/2}$ is the square root of the prior covariance matrix C_U. In our work we use $C_U^{1/2} = L$ where $C_U = LL^T$ is the Cholesky decomposition of C_U. Therefore, $\widehat{u}_j^\ell$ is a sample from the Gaussian $N(u^\ell, C_U)$. The perturbations (u_j^ℓ's) are used to generate an approximate gradient of $C_U \nabla J(u^\ell)$ where for simplicity we use $J(u)$ for the net present value function $J[u, m, y(u, m)]$ defined in Eq. 1.1. This approximation comes about by first approximating the cross covariance between the control variable $u \sim N(u^\ell, C_U)$ and the net present value $J(u)$ represented by the set of values $J(\widehat{u}_j^\ell), j = 1, 2, \ldots, N_e$. This cross-covariance matrix is denoted by C_{U^ℓ, J^ℓ} and defined by

$$C_{U,J}^\ell = \operatorname{cov}(u^\ell, J(u^\ell)) \approx \frac{1}{N_e - 1} \sum_{j=1}^{N_e} (\widehat{u}_j^\ell - \overline{\widehat{u}}^\ell)(J(\widehat{u}_j^\ell) - \overline{J}^\ell)^T, \tag{4.2}$$

where the mean of control variables appearing in Eq. 4.2 is defined as

$$\overline{\widehat{u}}^{\ell} = \frac{1}{N_e} \sum_{j=1}^{N_e} \widehat{u}_j^{\ell}, \tag{4.3}$$

and the mean of the NPV is defined as

$$\overline{J}^{\ell} = \frac{1}{N_e} \sum_{j=1}^{N_e} J(\widehat{u}_j^{\ell}). \tag{4.4}$$

Somewhat similar to the work of Reynolds et al. (2006), who showed the ensemble Kalman filter update is similar to one Gauss-Newton iteration using an average sensitivity matrix to update each ensemble member, Chen et al. (2009) derived a formula for ensemble-based optimization (EnOpt) by using the approximations of the following two equations:

$$\overline{J}^{\ell} \approx J(\overline{\widehat{u}}^{\ell}) \approx J(u^{\ell}), \tag{4.5}$$

$$J(\widehat{u}_j^{\ell}) \approx J(u^{\ell}) + \big(\nabla_u J(u^{\ell})\big)^T (\widehat{u}_j^{\ell} - u^{\ell}), \tag{4.6}$$

where Eq. 4.6 is simply a first-order Taylor series approximation. Moreover, Chen et al. (2009) observed that Eq. 4.1 implies that the mean of the sample of perturbed controls should be approximately equal to the estimate of the vector of optimal controls at the ℓ^{th} iteration , i.e.,

$$\overline{\widehat{u}}^{\ell} \approx u^{\ell}. \tag{4.7}$$

Using the approximations of Eqs. 4.5, 4.6 and 4.7 in Eq. 4.2 gives

$$C_{U,J}^{\ell} = \frac{1}{N_e - 1} \sum_{j=1}^{N_e} (\widehat{u}_j^{\ell} - \overline{\widehat{u}}^{\ell})(\widehat{u}_j^{\ell} - \overline{\widehat{u}}^{\ell})^T \nabla_u J(u^{\ell}) \approx C_U \nabla_u J(u^{\ell}). \tag{4.8}$$

Thus, the cross-covariance between the controls and the NPV function is approximately equal to C_U times the gradient of NPV, i.e., represents a preconditioned steepest ascent direction or Newton-type method with C_U used to approximate the inverse Hessian. Instead of using this preconditioned steepest descent direction as the search direction, however, Chen et al. (2009), multiplied again by C_U for additional smoothing to obtain the search direction

$$d^\ell = C_U C_{U,J}^\ell \approx C_U^2 \nabla_u J(u^\ell). \tag{4.9}$$

Then the following equation is used to update control variables

$$u^{\ell+1} = u^\ell + \alpha^\ell d^\ell = u^\ell + \alpha^\ell C_U C_{U,J}^\ell, \tag{4.10}$$

where α^ℓ is the step size. If $J(u^{\ell+1}) \leq J(u^\ell)$, then $u^{\ell+1}$ is not accepted as the new estimate of the vector of optimal controls; instead we decrease the step-size and reapply Eq. 4.10.

Chen et al. (2009) provided no information on how to initialize the stepsize α_0^ℓ for each iteration or perhaps more importantly what to do if d^ℓ is not an uphill direction. Because Eqs. 4.8 and 4.9 are approximate, the search direction d^ℓ given in Eq. 4.9 is not always an uphill direction. When Eq. 4.9 gives a downhill direction, cutting the step size cannot be expected to finally arrive at a control vector which will increase the value of the NPV function, J. To resolve these issues, we use a modified implementation presented in the following paragraph. This modification is used in all results presented in this study.

Our implementation of EnOpt is as follows: First replace Eq. 4.10 by

$$u^{\ell+1} = u^\ell + \alpha_\ell \frac{C_U C_{U,J}^\ell}{\parallel C_U C_{U,J}^\ell \parallel_\infty}. \tag{4.11}$$

After normalization of the search direction, we set the initial stepsize α_0^ℓ to be approximately equal to around 1/10 of the length of the smallest interval specifying bounds on a component of the control vector, i.e.,

$$\alpha_0^\ell = \frac{1}{10} \min_{1 \le i \le n_u} \{u_i^{\mathrm{up}} - u_i^{\mathrm{low}}\}. \tag{4.12}$$

For example, if bounds on well-bore pressure controls in psi are $1500 \le p_{wf} \le 3500$, and the bounds on the liquid flow rate in STB/D at each well are $0 \le q_\ell \le 10,000$, then an appropriate value of α_0^ℓ would be 200. The choice of 1/10 allow the algorithm to fully explore the range of feasible controls in a reasonable number of iterations. If at 10 subsequent iterations $\parallel C_U C_{U,J}^\ell \parallel_\infty$, were given by the same component of $C_{U,J}^\ell$ then it would be possible to reach the closest upper or lower bound on the corresponding wellbore pressure components of the control vector in ten or fewer iterations. As the maximum component of $C_{U,J}^\ell$ may vary from iteration to iteration, it will of course typically take far more iterations than ten to reach a bound. The examples considered here use the initial value of the step size given in Eq. 4.12. Although this choice is just a rule of thumb, it has proved to be a good choice for the examples we have done. If $\min_{1\le i\le n_u}[u_i^{\mathrm{up}} - u_i^{\mathrm{low}}] \ll \max_{1\le i\le n_u}[u_i^{\mathrm{up}} - u_i^{\mathrm{low}}]$, it may be preferable to rescale variables by replacing each u_i by

$$\widehat{u}_i = \frac{u_i - u_i^{\mathrm{low}}}{u_i^{\mathrm{up}} - u_i^{\mathrm{low}}}, \ i = 1, 2, \ldots, n_{\mathrm{u}}. \tag{4.13}$$

With this rescaling the bounding interval for each $\widehat{u}_i$ is [0,1]. For the problems considered here, we have not applied the rescaling of Eq. 4.13.

In the examples presented here, we simply define C_U from a spherical covariance function with specified variance and correlation "length" which are defined for each case considered in the example section. This covariance function is applied on a well by well basis so there is no correlation between controls at any two well pairs. If with $\alpha_\ell = \alpha_0^\ell$, the $u^{\ell+1}$ computed from Eq. 4.11 does not increase the NPV, then the step size is cut in half and $u^{\ell+1}$ is recomputed with this reduced step size. This process of cutting the step size continues until we have obtained an increase in the NPV or we have performed the maximum number of step size reductions that are allowable, which is five in the examples presented here. If with the maximum number of stepsize cuts, a $u^{\ell+1}$ has not been found such that $J(u^{\ell+1}) > J(u^{\ell})$, then we generate a new set of perturbed controls from Eq. 4.1 for use in Eq. 4.2 to try to find an uphill search direction. If after five successive perturbations, an uphill direction is not found, then the algorithm is terminated. This is our first termination or "convergence" criterion.

We also terminate the algorithm whenever the relative increase in the objective function is less than 10^{-4}, i.e.,

$$\frac{|J(u^{\ell+1}) - J(u^{\ell})|}{J(u^{\ell})} \leq 10^{-4}, \tag{4.14}$$

and the ℓ_2 norm of the relative change in the control vector is less than 10^{-3}, i.e.,

$$\frac{\|u^{\ell+1} - u^{\ell}\|}{\max(\|u^{\ell}\|, 1.0)} \leq 10^{-3}. \tag{4.15}$$

This is our second termination criterion.

As the third termination criterion, we specify the number of maximum allowable simulation runs as 2000 in the examples presented in this work. For all examples presented in the paper, our modification of EnOpt was terminated by the first criterion. In fact, by adding this termination criterion as opposed to using only the second and third termination criteria, we reduced the number of iterations required for convergence by more that 50% in all cases with negligible change in the estimated optimal NPV.

4.2 Simplex Gradient (SG) Method

Simplex gradients (SG) are basically the first order coefficients of polynomial interpolation or regression models, which, in turn, are used in derivative-free trust region methods. However, simplex gradients (SG) were used by Bortz and Kelley (1998) in their implicit filtering method, which can be viewed as a line search method based on simplex gradients. Tseng (1999) developed a class of simplex-based direct search methods imposing sufficient decrease conditions. He suggested the use of the norm of a simplex gradient in a stopping criterion for his class of methods. No numerical results were reported with this criterion, and no other use of the simplex gradient was suggested. In the context of the Nelder – Mead simplex-based direct search algorithm, Kelley (1999) used the simplex gradient norm in a sufficient decrease-type condition to detect stagnation, and the simplex gradient signs to orient the simplex restarts.

Calculation of a simplex gradient first requires the selection of a set of sample points. The geometrical properties of the sample set determine the quality of the corresponding simplex gradient as an approximation to the exact gradient of the objective function. As discussed below, Custsódio and Vicente (2007) determined simplex gradients under general conditions.

To obtain a unique simplex gradient of $J(u)$ where u is the n_u-dimension column vector of control variables, we must evaluate the objective function at exactly $n_u + 1$ independent points. The convex hull of a set of $n_u + 1$ affinely independent points $\{u^\ell, \widehat{u}_1^\ell, \ldots, \widehat{u}_{n_u}^\ell\}$ is called a simplex. The $n_u + 1$ points are the vertices of the simplex. Since the points are affinely independent, the $n_u \times n_u$ matrix $\Delta U = [\widehat{u}_1^\ell - u^\ell, \ldots, \widehat{u}_{n_u}^\ell - u^\ell]$ is nonsingular. Given a simplex of vertices $u^\ell, \widehat{u}^\ell, \ldots, \widehat{u}_{n_u}^\ell$, the simplex gradient at u^ℓ is defined as $\nabla_u J(u^\ell) = \Delta U^{-T}(\Delta J(u^\ell))^T$, with $\Delta J(u^\ell) = [J(\widehat{u}_1^\ell) - J(u^\ell), \ldots, J(\widehat{u}_{n_u}^\ell) - J(u^\ell)]$.

The simplex gradient is intimately related to linear multivariate polynomial interpolation. In fact, letting $\overline{g}$ denote the simplex gradient, it is easy to see that the linear model $m(u) = J(u^\ell) + \overline{g}^T(u - u^\ell)$ centered at u^ℓ interpolates J at the points $\widehat{u}_1^\ell, \ldots, \widehat{u}_{n_u}^\ell$.

In practical production optimization problems, n_u is large and evaluation of $J(u^\ell)$ requires a reservoir simulation run so it is not computationally feasible to evaluate J at $n_u + 1$ points. Thus, we must generate a simplex gradient using N_e points where $N_e \ll n_u$, one might have $N_e+1 \neq n_u+1$ points from which to compute a simplex gradient. We say that a sample set is poised for a simplex gradient calculation if ΔU is full rank, i.e., if $\text{rank}(\Delta U) = \min\{N_e, n_u\}$. (The notions of poisedness and affine independence coincide for $N_e < n_u$). Given the sample set $\{u^\ell, \widehat{u}_1^\ell, \ldots, \widehat{u}_{N_e}^\ell\}$, the simplex gradient $\nabla J(u^\ell)$ of J at u^ℓ can be defined as the solution $\overline{g}$ of the system

$$(\Delta U^\ell)^T \overline{g} = (\Delta J^\ell)^T, \tag{4.16}$$

where

$$\Delta U^\ell = [\widehat{u}_1^\ell - u^\ell, \ldots, \widehat{u}_{N_e}^\ell - u^\ell], \tag{4.17}$$

and

$$\Delta J^\ell = [J(\widehat{u}_1^\ell) - J(u^\ell), \ldots, J(\widehat{u}_{N_e}^\ell) - J(u^\ell)]. \tag{4.18}$$

Eq. 4.16 can be solved by singular value decomposition (SVD) solution. The SVD solution is the minimum norm solution if $N_e < n_u$.

We can avoid using the pseudo-inverse to solve Eq. 4.16 by using a preconditioned simplex gradient is defined by

$$\begin{aligned} C_U \overline{g} &\approx \frac{1}{N_e - 1}(\Delta U^\ell)(\Delta U^\ell)^T \overline{g} = \frac{1}{N_e - 1} \Delta U^\ell (\Delta U^\ell)^T (\Delta U^\ell)^{-T} (\Delta J^\ell)^T \\ &= \frac{1}{N_e - 1}(\Delta U^\ell)(\Delta J^\ell)^T = \frac{1}{N_e - 1} \sum_{j=1}^{N_e} (\widehat{u}_j^\ell - u^\ell)(J(\widehat{u}_j^\ell) - J(u^\ell)). \end{aligned} \tag{4.19}$$

Therefore, assuming again that $\overline{\widehat{u}} \approx u^\ell$ and $\overline{J} \approx J(u^\ell)$, and using Eq. 4.6, the search direction based on a preconditioned simplex gradient is given by

$$g_{ps}^{\ell} \equiv \frac{1}{N_e - 1} \sum_{j=1}^{N_e} (\widehat{u}_j^{\ell} - \overline{\widehat{u}}^{\ell})(\widehat{u}_j^{\ell} - \overline{\widehat{u}}^{\ell})^T \nabla_u J(u^{\ell}) \approx C_U \nabla_u J(u^{\ell}). \tag{4.20}$$

As an important comment, we note that we divided by $N_e - 1$ in Eq. 4.19 so that Eq.4.20 has exactly the same the form as Eq. 4.8 for EnOpt.

Similar to the EnOpt algorithm, we can multiply the search direction in Eq. 4.20 by C_U for additional smoothing to obtain the search direction given by

$$d^{\ell} = C_{U^{\ell}} g_{ps}^{\ell} \approx C_{U^{\ell}}^2 \nabla_u J(u^{\ell}). \tag{4.21}$$

We also normalize the search direction to update the control variables, i.e., iteration of the algorithm are denoted by

$$u^{\ell+1} = u^{\ell} + \alpha^{\ell} \frac{C_{U^{\ell}} g_{ps}^{\ell}}{\parallel C_{U^{\ell}} g_{ps}^{\ell} \parallel_{\infty}}, \tag{4.22}$$

where α^{ℓ} is stepzise. The procedure to choose an initial stepzise α_0^{ℓ} is identical to that using in the EnOpt algorithm. The convergence criteria in Eq. 4.22 are identical to the three termination conditions used in EnOpt.

4.3 The Theoretical Connection Between Derivative-Free Algorithms Using Easily Calculated Approximations to The Gradient or The Preconditioned Gradient

The purpose of this section is to show that the preconditioned gradient $C_U \nabla J$ used in EnOpt, G-SPSA and SG are approximately the same and to explain the differences in the application of the associated optimization algorithms.

Reynolds et al. (2006) showed that EnKF is approximately equivalent to applying one iteration of the Gaussian-Newton method with the same average sensitivity matrix to update each ensemble member with the ensemble member used as the initial guess in the Gauss-Newton algorithm. We will use the same type of analysis to show that the basis EnOpt gradient, the simplex gradient and the SPSA gradient based on Gaussian perturbation represent three different approximations of the preconditioned gradient $C_U \nabla J(u^\ell)$

To define the preconditioned simplex method, we start with a more general approach. We generate N_e independent samples, $\delta \widehat{u}_j^\ell$, $j = 1, \cdots, N_e$ from the multivariate Gaussian distribution $N(0, C_U^\ell)$ and define

$$\widehat{u}_j^\ell = u^\ell + \delta \widehat{u}_j, \; j = 1, \ldots, N_e, \tag{4.23}$$

so $\widehat{u}_j^\ell \sim N(u^\ell, C_U^\ell)$. Define

$$\delta J_j^\ell = J(u^\ell + c_\ell \delta \widehat{u}_j) - J(u^\ell), \tag{4.24}$$

where c_ℓ is a constant but can change with the iteration index ℓ.

For simplicity in notation, we assume $J(u)$ is a position definite quadratic with constant real symmetric positive definite matrix H. This assumption simply avoids truncation in the following second order Taylor series

$$\delta J_j^\ell = J(u^\ell + c_\ell \delta \widehat{u}_j) - J(u^\ell) = c_\ell \nabla J(u^\ell)^T \delta \widehat{u}_j^\ell + \frac{1}{2} c_\ell^2 (\delta \widehat{u}_j^\ell)^T H (\delta \widehat{u}_j^\ell), \tag{4.25}$$

which holds for $j = 1, \ldots, N_e$. Defining

$$\Delta J^\ell = [\delta J_1^\ell, \ldots, \delta J_{N_e}^\ell], \tag{4.26}$$

it follows that

$$\Delta J^\ell = c_\ell (\nabla J(u^\ell))^T [\delta \widehat{u}_1^\ell, \delta \widehat{u}_2^\ell, \ldots, \delta \widehat{u}_{N_e}^\ell] + c_\ell^2 e^\ell, \tag{4.27}$$

where

$$e^\ell = \frac{1}{2}[(\delta\widehat{u}_1^\ell)^T H \delta\widehat{u}_1^\ell, \ldots, (\delta\widehat{u}_{N_e}^\ell)^T H \delta\widehat{u}_{N_e}^\ell]. \tag{4.28}$$

Defining $\delta\widehat{U}^\ell$ by

$$\delta\widehat{U}^\ell = [\delta\widehat{u}_1^\ell, \delta\widehat{u}_2^\ell, \ldots, \delta\widehat{u}_{N_e}^\ell]. \tag{4.29}$$

Eq. 4.27 becomes

$$\Delta J^\ell = c_\ell(\nabla J(u^\ell))^T \delta\widehat{U}^\ell + c_\ell^2 e^\ell. \tag{4.30}$$

Taking the transpose of Eq. 4.30 and then multiplying by $\frac{1}{N_e}(\delta\widehat{U}^\ell)$ gives

$$\frac{1}{N_e}(\delta\widehat{U}^\ell)(\Delta J^\ell)^T = c_\ell \frac{1}{N_e}(\delta\widehat{U}^\ell)(\delta\widehat{U}^\ell)^T(\nabla J(u^\ell)) + c_\ell^2 h^\ell, \tag{4.31}$$

where $h^\ell = \frac{1}{N_e}(\delta\widehat{U}^\ell)(e^\ell)^T$.

We write Eq. 4.31 as

$$\frac{1}{N_e}\sum_{j=1}^{Ne}(\delta\widehat{u}_j^\ell)[J(u^\ell + c_\ell\delta\widehat{u}_j^\ell) - J(u^\ell)] = \left[\frac{1}{N_e}\sum_{j=1}^{N_e} c_\ell(\delta\widehat{u}_j^\ell)(\delta\widehat{u}_j^\ell)^T\right]\nabla J(u^\ell) + c_\ell^2 h^\ell, \tag{4.32}$$

where we used the fact that δJ_j^ℓ is a scalar so $(\delta J_j^\ell)^T = \delta J_j^\ell$.

In the preconditioned simplex gradient, we actually set $c_\ell = 1$, so Eq. 4.32 becomes

$$\overline{g}_{ps}^\ell = \frac{1}{N_e}\sum_{j=1}^{N_e}\delta\widehat{u}_j^\ell[J(u^\ell + \delta\widehat{u}_j^\ell) - J(u^\ell)] = \frac{1}{N_e}\sum_{j=1}^{N_e}(\delta\widehat{u}_j^\ell)(\delta\widehat{u}_j^\ell)^T \nabla J(u^\ell)] + h^\ell, \tag{4.33}$$

where the first equality of Eq. 4.33 defines the preconditioned simplex gradient at the ℓ^{th} iteration which is denoted by $\overline{g}_{ps}^\ell$.

Since $E[\delta\widehat{u}_j^\ell(\delta\widehat{u}_j^\ell)^T] = C_U^\ell$, taking expectation in Eq. 4.33 gives

$$E[\overline{g}_{ps}^\ell] = C_{U^\ell}\nabla J(u^\ell) + E[h^\ell]. \tag{4.34}$$

The error term h^ℓ is given by

$$h^\ell = \frac{1}{2N_e}\delta\widehat{U}^\ell \begin{bmatrix} (\delta\widehat{u}_1^\ell)^T H\delta\widehat{u}_1^\ell \\ (\delta\widehat{u}_2^\ell)^T H\delta\widehat{u}_2^\ell \\ . \\ . \\ (\delta\widehat{u}_{N_e}^\ell)^T H\delta\widehat{u}_{N_e}^\ell \end{bmatrix} = \frac{1}{2N_e}\sum_{j=1}^{N_e}\delta\widehat{u}_j^\ell\left[(\delta\widehat{u}_j^\ell)^T H\delta\widehat{u}_j^\ell\right]. \tag{4.35}$$

As all $\delta\widehat{u}_j^\ell \sim N(0, C_U^\ell)$, $E[\delta\widehat{u}_j^\ell] = 0$, so $\|h^\ell\|$ may in practice be small but there is no way to show this term goes to zero as $\ell \to \infty$.

Recall from Eq. 4.23 that

$$\delta\widehat{u}_j = \widehat{u}_j^\ell - u^\ell,\ j = 1, \ldots, N_e, \tag{4.36}$$

so we may write the simplex gradient as

$$\overline{g}_{ps}^\ell = \frac{1}{N_e}\sum_{j=1}^{N_e}(\widehat{u}_j^\ell - u^\ell)(J(\widehat{u}_j^\ell) - J(u^\ell)). \tag{4.37}$$

Under the additional assumption that

$$\overline{\widehat{u}}^\ell = \frac{1}{2N_e}\sum_{j=1}^{N_e}\widehat{u}^\ell \approx u^\ell, \tag{4.38}$$

where is reasonable and that

$$\overline{J}^\ell = \frac{1}{N_e}\sum_{j=1}^{N_e}J(\widehat{u}_j^\ell) \approx J(u^\ell), \tag{4.39}$$

which is reasonable if J is not too nonlinear. Eq. 4.37 can be accurately approximated by

$$\overline{g}^{\ell}_{ps} = \frac{1}{N_e} \sum_{j=1}^{N_e} (\widehat{u}^{\ell}_j - \overline{\widehat{u}}^{\ell})(J(\widehat{u}^{\ell}_j) - \overline{J}^{\ell}), \tag{4.40}$$

but the basic EnOpt gradient (Wang et al., 2009) is

$$\overline{g}^{\ell}_{EnOpt} = \frac{1}{N_e - 1} \sum_{j=1}^{N_e} (\widehat{u}^{\ell}_j - \overline{\widehat{u}}^{\ell})(J(\widehat{u}^{\ell}_j) - \overline{J}^{\ell}) \approx C_{U^{\ell}J^{\ell}} = \frac{N_e}{Ne - 1} \overline{g}^{\ell}_{ps}. \tag{4.41}$$

The reason $1/(N_e - 1)$ was used in Eq. 4.41 is that the second term in Eq. 4.41 is the standard estimate of the cross-covariance matrix $C_{U^{\ell}J^{\ell}}$. Note in Eq. 4.41, we estimate the mean $\overline{\widehat{u}}^{\ell}$ from the sample, whereas in Eq. 4.37 we did not estimate a mean. Nevertheless, the results of Eq. 4.41 indicate that the basic EnOpt and the preconditioned simplex gradient both to approximate in expectation $C_{U^{\ell}}\nabla(u^{\ell})$, the gradient of J preconditioned by the covariance matric $C_{U^{\ell}}$. Although the error term in this two methods is slightly different, there appears to be no way to guarantee that this error term goes to zero as $\ell \to \infty$.

We now divide Eq. 4.32 by c_{ℓ} and write the resulting equation as

$$\frac{1}{N_e} \sum_{j=1}^{Ne} \left[\frac{J(u^{\ell} + c_{\ell}\delta\widehat{u}^{\ell}_j) - J(u^{\ell})}{c_{\ell}} \right] (\delta\widehat{u}^{\ell}_j) = \left[\frac{1}{N_e} \sum_{j=1}^{N_e} (\delta\widehat{u}^{\ell}_j)(\delta\widehat{u}^{\ell}_j)^T \right] \nabla J(u^{\ell}) + c_{\ell}h^{\ell}. \tag{4.42}$$

Note that left side of Eq. 4.42 represents the average of N_e SPSA gradients generated by Gaussian perturbations, i.e.,

$$\overline{\widehat{g}}(u^{\ell}) = \frac{1}{N_e} \sum_{j=1}^{Ne} \left[\frac{J(u^{\ell} + c_{\ell}\delta\widehat{u}^{\ell}_j) - J(u^{\ell})}{c_{\ell}} \right] (\delta\widehat{u}^{\ell}_j). \tag{4.43}$$

From Eq. 4.42 and the fact that $E[\delta\widehat{u}^{\ell}_j(\delta\widehat{u}^{\ell}_j)^T] = C_{U^{\ell}}$, we got

$$E[\overline{\widehat{g}}(u^{\ell})] = C_{U^{\ell}}\nabla J(u^{\ell}) + c_{\ell}E[h^{\ell}]. \tag{4.44}$$

Now assuming $E[h^\ell]$ is uniformly bounded independent of ℓ and we require $\lim_{\ell\to\infty} c_\ell = 0$ it follow that

$$\lim_{\ell\to\infty} E[\overline{\widehat{g}}(u^\ell)] = C_{U^\ell}\nabla J(u^\ell), \tag{4.45}$$

thus the SPSA algorithm has a theoretically attractive property that EnOpt and steepest ascent with a preconditioned simplex gradient do not have.

4.4 Numerical Comparison of Algorithms

4.4.1 Production Optimization with simple bound constraints for a three-channel reservoir

In this example, we use the same reservoir model and well data as example 1 of the Chapter 1 (Fig. 4.1). It means that there are a total of 13 vertical wells with 4 production wells and 9 injection wells arranged in a five-spot well pattern. The anticipated total project life is 1800 days and the control step size is set equal to 180 days so we have 10 control steps. The total number of control variables is $(4 + 9) \times 10 = 130$. To optimize the NPV, we set each injection well under water injection rate control with a lower bound of 0 STB/D and an upper bound of 2000 STB/D, and each production well under bottom hole pressure (BHP) control with a lower bound of 1500 psi and an upper bound of 6000 psi. The oil price is set at \$50/STB, the water injection rate cost at \$0/BBL, the water production cost at \$5.56/STB, and the annual discount rate is 0%.

The initial guess for the water injection rate control of each injection well is equal to 300 STB/D, and the initial guess for BHP control of each production well is equal to 3500 psi. As always to deal with the bound constraints, we use the log-transformation to convert the bounded control variables into unbounded control variables, i.e., the constrained problem is turned into an unconstrained problem.

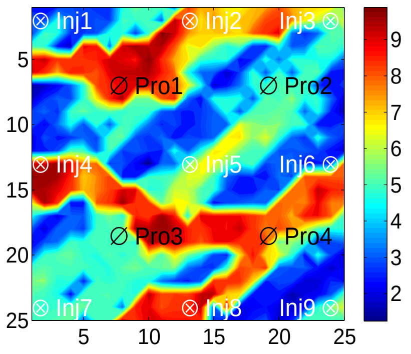


Figure 4.1: ln(k) distribution.

For the G-SPSA algorithm, we consider $M = 5$ and $M = 10$ where M denotes the number of G-SPSA gradients (Eq. 2.17) to approximate the steepest ascent direction in order to update the control variables (see 2.21). The control variables are sampled from Gaussian distribution with variances $\sigma^2 = (1, 1)$, of which the first number corresponds to the transformed variable of BHP control and the second number corresponds to the transformed variable of water injection rate control, and two correlation lengths of $N_s = 6$ and $N_s = 10$ (Eq. 2.15). The values of α and γ in these equations are similar to the values suggested by Spall (1998), $\alpha = 0.602$ and $\gamma = 0.101$. We set the maximum number of allowable iterations equal to 200, which is almost two times the number of control variables (130), then by the general procedure in Section 2.3 we obtain $A = 20$. By setting the value of a_0 equal to 2.0, we obtain $a = 13.0$. We set c_{min} equal to 0.0585 which gives $c = 0.1$. The condition for terminating the G-SPSA algorithm is based on the maximum number of allowable iterations.

In addition, in this example, we use the smooth G-SPSA (SmG-SPSA) algorithm to optimize the NPV function. The SmG-SPSA gradient is multiplied the G-SPSA gradient by covariance matrix C_U (Eq. 2.15), i.e., the search direction of SmG-SPSA is approximately equal to $C_U^2 \nabla J(u^\ell)$. Other parameters used in SmG-

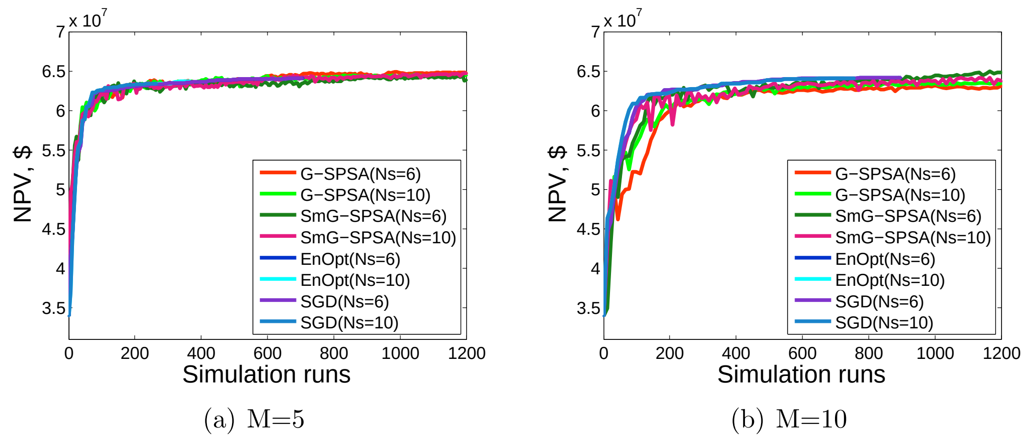


Figure 4.2: NPV versus the number of simulation runs obtained from one optimization run of each algorithm with one initial random seed ($\sigma^2 = (1, 1)$; SPSA parameters: $k_{\max} = 200$, $a_0 = 2.0$, $c = 0.1$; EnOpt parameters: $a_0 = 1.0$; SG parameters: $a_0 = 1.0$).

SPSA are the same as used in the G-SPSA algorithm. The condition for terminating SmG-SPSA is also based on the maximum number of allowable iterations.

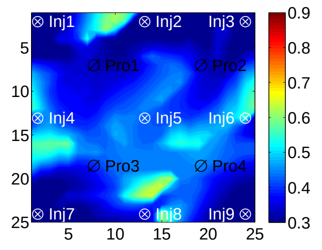
For the EnOpt and the SG (Simplex gradient) algorithms, we use an ensemble size of $N_e = 5$ and $N_e = 10$ to generate an approximate gradient. To construct the covariance C_U, we use the variance $\sigma^2 = (1, 1)$, of which the first number corresponds to the transformed variable of BHP control and the second number corresponds to the transformed variable of water injection rate control, and two correlation lengths of $N_s = 6$ and $N_s = 10$ (Eq. 2.15). The initial step size is equal to 1. The normalized search directions in Eqs. 4.11 and 4.22 are used to update the control variables for the EnOpt algorithm and the SG algorithm, respectively. The condition for terminating EnOpt and SG, respectively, is based on discussion on Section 4.1 and Section 4.2.

To compare the performance of different algorithms, all algorithms have been tested with five different initial random seeds. The average NPV and average simulation run calculated by five different optimization runs will be compared against other results. Recall that when $M = 5$, we require 6 reservoir simulation runs per iteration but with $M = 10$ we require 11 reservoir simulation runs per iteration. However, the

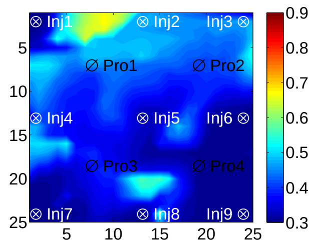
NPV obtained from the G-SPSA and SmG-SPSA algorithms at the 1200^{th} simulation run is used to compare against others NPV's obtained from EnOpt and SG which stop at different numbers of simulation runs.

Fig. 4.2 presents the NPV versus number of simulation run for all algorithms when we change the value of M and correlation length N_s. Note that this figure only present the relationship between the NPV and simulation obtained from one optimization with one initial random seed. From Fig. 4.2, we can see that when the value of M is equal to 5, the final NPV and the convergence rate obtained from all algorithms are similar to each other, even when the correlation lengths N_s are set to different values. Meanwhile, when the value of M is equal to 10, the final NPV and the simulation runs obtained from the EnOpt and SG algorithms, which have been tested with two values of correlation length, are almost similar to each other. However, these NPV's and convergence rates are better than those obtained from the G-SPSA algorithm, which have also been tested with two different correlation lengths. This may be because in SG and EnOpt we smooth twice, i.e., the search direction is approximately equal to $C_U^2 \nabla J(u^\ell)$ whereas in G-SPSA, it is approximately $C_U \nabla J(u^\ell)$. However, convergence rates obtained from SmG-SPSA are less than those obtained from the SG and EnOpt algorithm even though the search direction of SmG-SPSA is also approximately equal to $C_U^2 \nabla J(u^\ell)$.

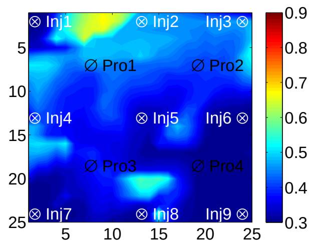
Similar to the final NPV and convergence rate obtained from all algorithms, which is analyzed in the previous step, the final remaining oil saturation distributions after optimization look very similar each other (Fig. 4.3). The final controls for the producers and injectors obtained by different algorithms are shown in Figs 4.4 and 4.5. Note that even through the final NPV and the remaining oil distribution obtained by different algorithms are not very different, the estimates of the optimal controls (Figs 4.4 and 4.5) are substantially different. This again suggests that sometimes there are multiple sets of controls which can achieve essentially the same



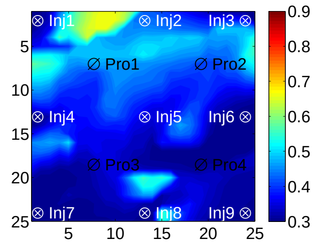
(a) G-SPSA($N_s = 6$)



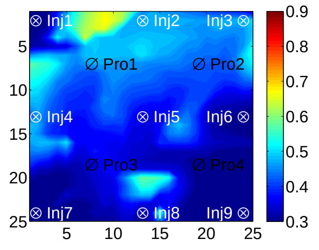
(b) EnOpt($N_s = 6$)



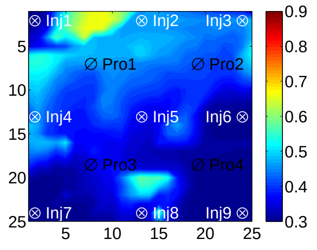
(c) SGD($N_s = 6$)



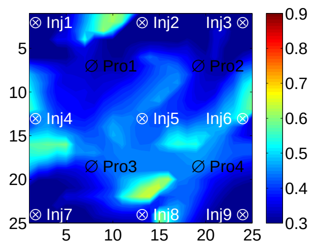
(d) G-SPSA($N_s = 10$)



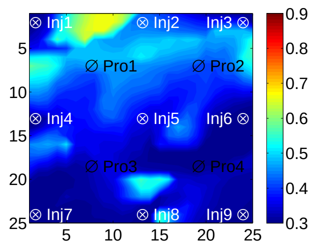
(e) EnOpt($N_s = 10$)



(f) SGD($N_s = 10$)



(g) SmG-SPSA($N_s = 6$)

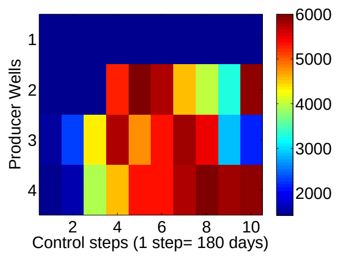


(h) SmG-SPSA($N_s = 10$)

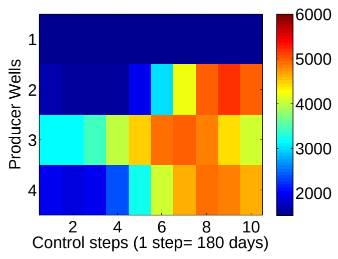
Figure 4.3: Remaining oil saturation distribution obtained from one optimization run of each algorithm ($M = 5$, $\sigma^2 = (1, 1)$; SPSA parameters: $k_{\max} = 200$, $a_0 = 2.0$, $c = 0.1$; EnOpt parameter: $a_0 = 1.0$; SG parameter: $a_0 = 1.0$).

maximum value of NPV. Note that Sm-GSPSA, EnOpt and SG result in the smoother final controls than those obtained from G-SPSA because Sm-GSPSA, EnOpt and SG smooth twice. For all algorithms, when the correlation length increases, we obtain a smoother final control variable. However, the full correlation length leads to very smooth final control variables in the Sm-GSPSA, EnOpt and SG algorithms.

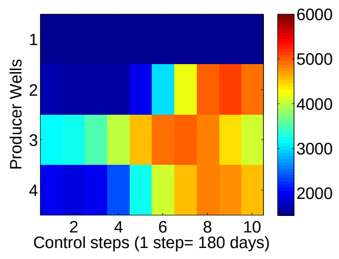
The performance of all algorithms is shown in Table 4.1. This table presents the average NPV and the average number of simulation runs of five different optimizations with five different initial random seeds. The final NPV obtained from the SmG-SPSA and G-SPSA algorithm is always higher than the corresponding NPV



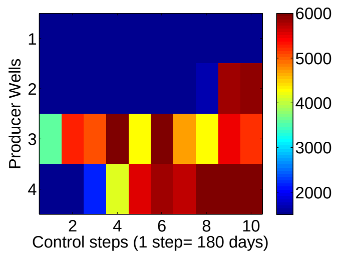
(a) G-SPSA($N_s = 6$)



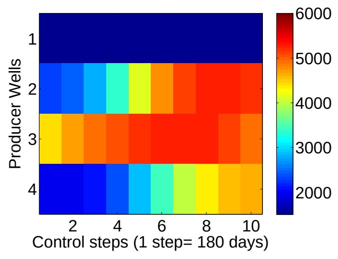
(b) EnOpt($N_s = 6$)



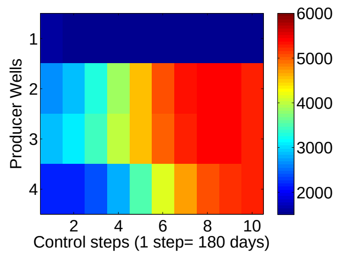
(c) SGD($N_s = 6$)



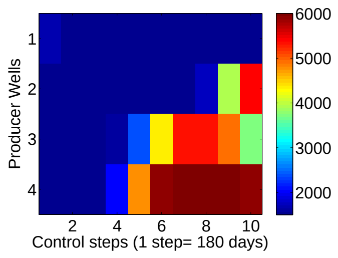
(d) G-SPSA($N_s = 10$)



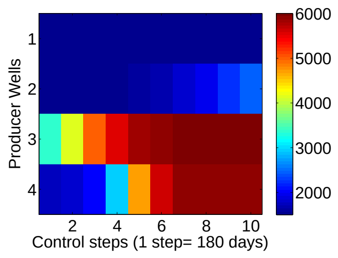
(e) EnOpt($N_s = 10$)



(f) SGD($N_s = 10$)



(g) SmG-SPSA($N_s = 6$)

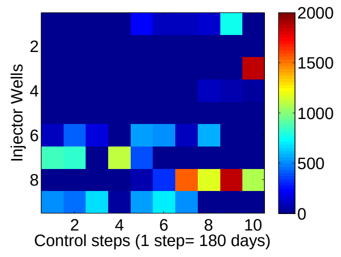


(h) SmG-SPSA($N_s = 10$)

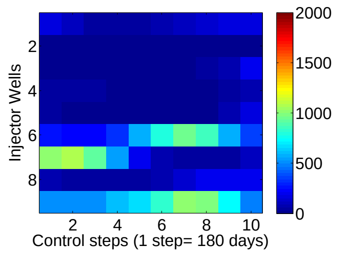
Figure 4.4: The estimated optimal BPH controls obtained from one optimization run of each algorithm ($M = 5$, $N_s = 6$ and $N_s = 10$, $\sigma^2 = (1, 1)$; SPSA parameters: $k_{\max} = 200$, $a_0 = 2.0$, $c = 0.1$; EnOpt parameters: $a_0 = 1.0$; SG parameters: $a_0 = 1.0$).

obtained from the EnOpt algorithm and the SG algorithm regardless of the value of M. However, the total required number of simulation runs of the G-SPSA algorithm is much more than those of the EnOpt and SG algorithms because it terminates only when the maximum number of allowable simulation runs is reached. The NPV and the total numbers of simulation runs obtained from the EnOpt algorithm are similar to those obtained from the SG algorithm. The final NPV obtained from the G-SPSA algorithm is always higher than that obtained from SmG-SPSA.

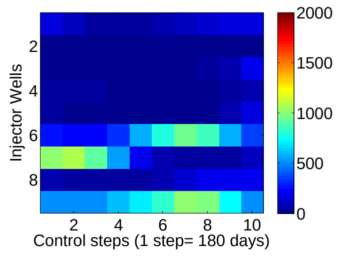
Finally, we summarize the results of the experimental computations.



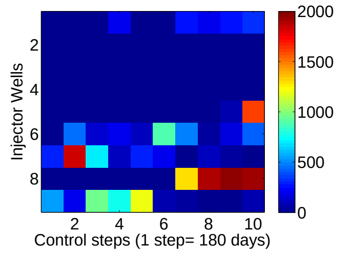
(a) G-SPSA($N_s = 6$)



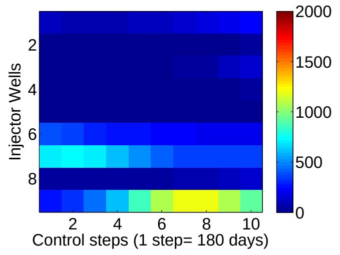
(b) EnOpt($N_s = 6$)



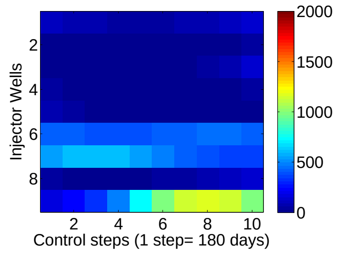
(c) SGD($N_s = 6$)



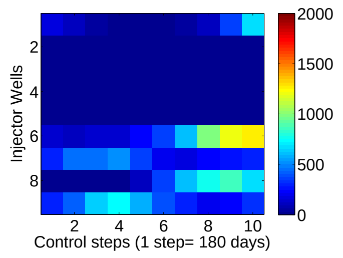
(d) G-SPSA($N_s = 10$)



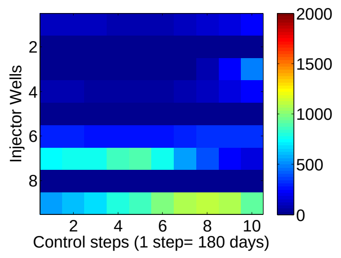
(e) EnOpt($N_s = 10$)



(f) SGD($N_s = 10$)



(g) SmG-SPSA($N_s = 6$)



(h) SmG-SPSA($N_s = 10$)

Figure 4.5: The estimated optimal water injection rate well controls obtained from one optimization run of each algorithm ($M = 5$, $N_s = 6$ and $N_s = 10$, $\sigma^2 = (1, 1)$; SPSA parameters: $k_{\max} = 200$, $a_0 = 2.0$, $c = 0.1$; EnOpt parameters: $a_0 = 1.0$; SG parameters: $a_0 = 1.0$).

- In this example, the EnOpt and SG algorithms always terminate because no further improvement can be made in NPV functional whereas the G-SPSA and SmG-SPSA algorithm terminated when the maximum number of iterations were reached. As a consequence, SmG-SPSA and G-SPSA always achieved a higher NPV but at the expense of more reservoir simulation runs. The final NPV's obtained from G-SPSA are always greater than those obtained from SmG-SPSA.

- The NPV's and the number of simulation runs obtained with the EnOpt algo-

Table 4.1: The performance of all algorithms obtained from five optimization runs of each algorithm with five different initial seeds when we change the values of M and N_s ($\sigma^2 = (1,1)$; SPSA parameters: $k_{\max} = 200$, $a_0 = 2.0$, $c = 0.1$; EnOpt parameters: $a_0 = 1.0$; SG parameters: $a_0 = 1.0$) .

Algorithm	$N_s = 6$		$N_s = 10$	
	Simulation	Final NPV	Simulation	Final NPV
G-SPSA($M = 5$)	1200	6.434×10^7	1200	6.433×10^7
SmG-SPSA($M = 5$)	1200	6.411×10^7	1200	6.424×10^7
EnOpt($M = 5$)	442	6.348×10^7	379	6.311×10^7
SGD($M = 5$)	468	6.355×10^9	363	6.307×10^7
G-SPSA($M = 10$)	1200	6.420×10^7	1200	6.427×10^7
SmG-SPSA($M = 10$)	1200	6.414×10^7	1200	6.394×10^7
EnOpt($M = 10$)	684	6.387×10^7	605	6.365×10^7
SGD($M = 10$)	724	6.386×10^7	582	6.362×10^7

rithm are very close to those obtained from the SG algorithm. The final well control are also fairly similar.

- Although the final NPV's and the remaining oil distributions obtained from different algorithms are not very different, the final control variables obtained from G-SPSA are significantly different than those obtained from other algorithms. This suggests that sometimes there are multiple sets of controls which can achieve essentially the same maximum value of NPV.

- In this example using a correlation length equal to the number of control steps, the SmG-SPSA, EnOpt and SG algorithms give very smooth final well controls and lower NPV than is obtained with a shorter correlation length.

4.4.2 Optimization with simple bound constraints for PUNQ case

We test all algorithms with the PUNQ case used the same model and data of Chapter 2 (Fig. 4.6). There are 7 vertical producing wells and 7 vertical water injection wells. The reservoir life is set equal to 7600 days. The length of each control step is set equal to 190 days. Thus, there are 40 control steps and the total number

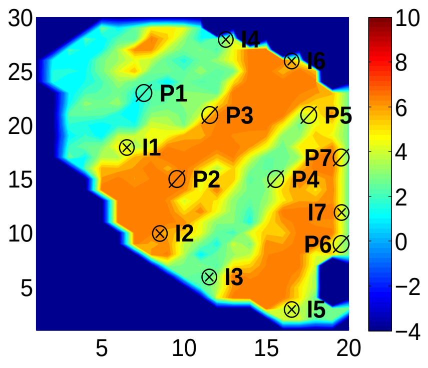
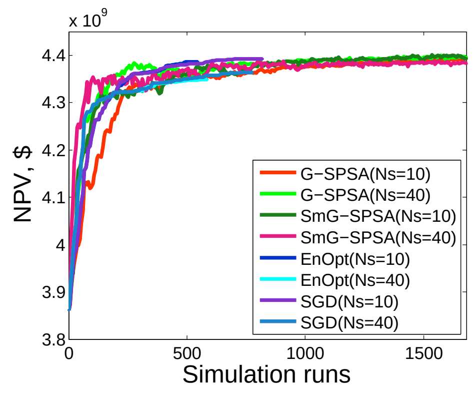


Figure 4.6: Well locations (Layer-1 horizontal log permeability field), Example 1.

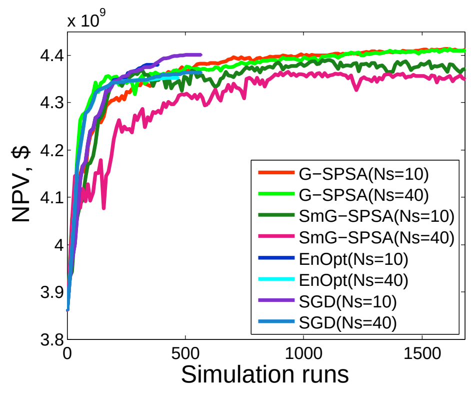
of control variables is equal to $(7+7)\times 40 = 560$. During optimization, the injection wells are placed under water injection rate control with a lower bound of 0 STB/D and an upper bound of 3000 STB/D. The production wells are placed under bottom hole pressure (BHP) control with a lower bound of 1000 psi and an upper bound of 3000 psi. The oil price is \$80.0/STB, the water production cost is \$8.9/STB, the water injection cost is \$0.0/STB, and the annual discount rate is 0.

The initial guess for the controls is equal to 1500 STB/D for each injection rate control and 1500 psi for each producer BHP control. The log-transformation is still applied to hanlde the bound constraints. As always in this work, the optimization is done in term of the transformed variables.

For G-SPSA and SmG-SPSA which smooths twice as multiplying the G-SPSA gradient by covariance matrix C_U (Eq. 2.15), the average of five ($M = 5$) and ten ($M = 10$) G-SPSA gradients (Eq. 2.17) is used to approximate the steepest ascent direction in order to update the control variables (Eq. 2.21). The control variables are sampled from Gaussian distribution with variances of $\sigma^2 = (1, 1)$, of which the first number corresponds to the transformed variable of BHP control and the second number corresponds to the transformed variable of water injection rate control, and two correlation lengths of $N_s = 10$ or $N_s = 40$ (Eq. 2.15). The values of α and γ



(a) M=5



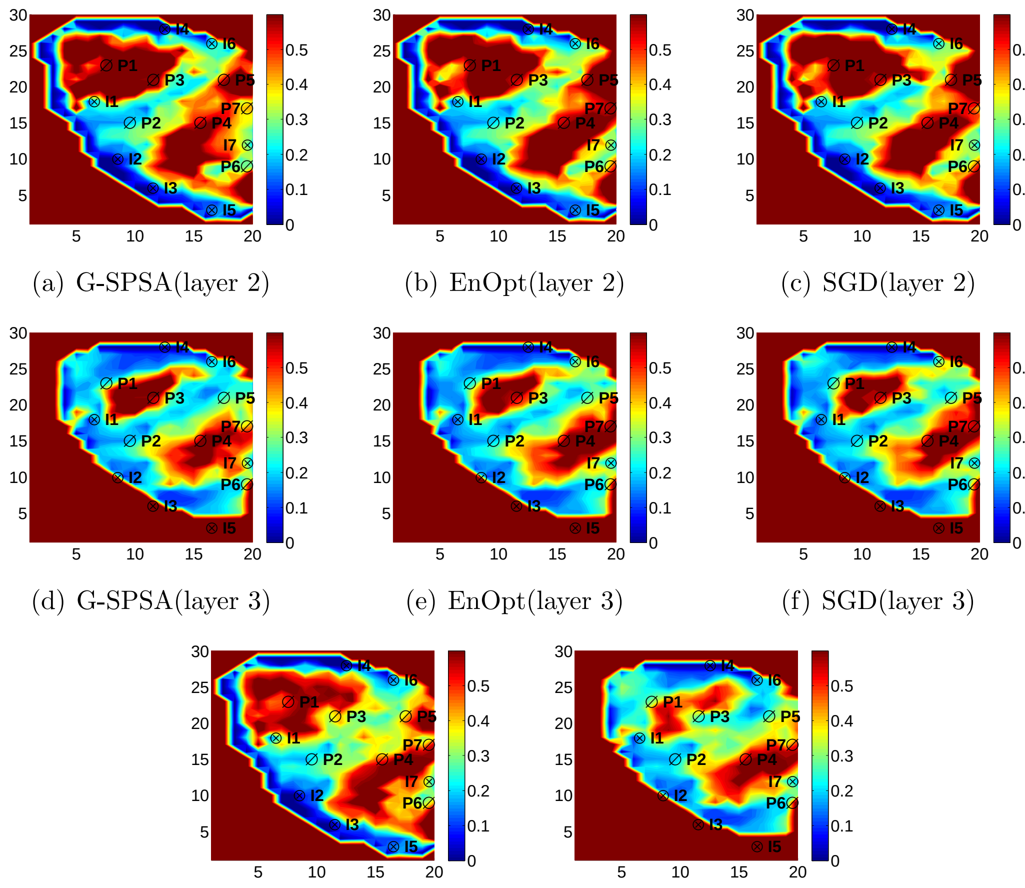
(b) M=10

Figure 4.7: NPV versus the number of simulation runs with different average numbers of SPSA ($\sigma^2 = (1, 1)$; SPSA parameters: $k_{\max} = 280$, $a_0 = 2.0$, $c = 0.1$; EnOpt parameters: $a_0 = 1.0$; SG parameters: $a_0 = 1.0$).

in these equations are identical to the values suggested by Spall (1998), $\alpha = 0.602$ and $\gamma = 0.101$. We set the maximum number of allowable iterations equal to 280, which is one half the number of control variables (560), we obtain $A = 28$. By setting the value of a_0 equal to 2.0, we obtain $a = 15.5$. We set c_{min} equal to 0.0585 which gives $c = 0.1$. The condition for terminating the G-SPSA and SmG-SPSA algorithms is based on the maximum number of allowable iteration. Note that when 5 SPSA gradients are used to calculate the average of SPSA gradient, 280 iterations correspond to 1680 reservoir simulation runs, and when 10 SPSA gradients are used, 200 iterations correspond to 3080 simulation runs.

For the EnOpt and SG algorithms, we use two ensemble sizes of $N_e = 5$ and $N_e = 10$ to generate an approximate gradients. The variance $\sigma^2 = (1, 1)$, of which the first number corresponds to the transformed variable of BHP control and the second number corresponds to the transformed variable of water injection rate control, and two correlation lengths of $N_s = 10$ or $N_s = 40$ are used to construct the covariance C_U (Eq. 2.15). The initial step size is equal to 1 and the normalized search directions in Eqs. 4.11 and 4.22 are used to update respectively the transformed variables for the EnOpt algorithm and the SG algorithm, respectively. The condition for terminating

EnOpt and SG, respectively, is based on discussion on Section 4.1 and Section 4.2.



(a) G-SPSA(layer 2) (b) EnOpt(layer 2) (c) SGD(layer 2)

(d) G-SPSA(layer 3) (e) EnOpt(layer 3) (f) SGD(layer 3)

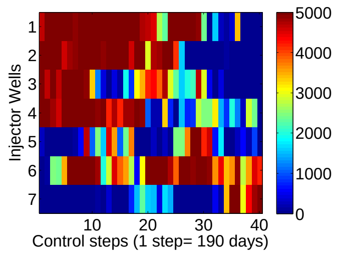
(g) SmG-GSPSA(layer 2) (h) SmG-GSPSA(layer 3)

Figure 4.8: Remaining oil saturation distribution obtained from one optimization run of each algorithm ($M = 5$, $\sigma^2 = (1, 1)$; SPSA parameters: $k_{\max} = 280$, $a_0 = 2.0$, $c = 0.1$; EnOpt parameters: $a_0 = 1.0$; SG parameters: $a_0 = 1.0$).

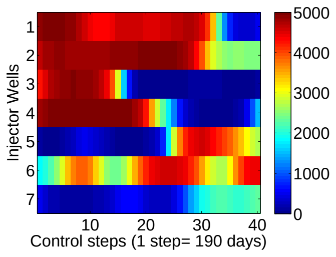
In this example, all algorithms also have been tested with five different initial random seeds. We compare the average NPV and number of simulation runs calculated by five different optimizations with five different initial random seeds. For the SmG-SPSA and G-SPSA algorithms, the NPV obtained at the 1680^{th} simulation run is used to compare with NPV's obtained from other algorithms.

Similar to the previous example in Subsection 4.4.1, we plot the NPV versus the number of simulation runs for all algorithms when we change the value of M (number of perturbations used to estimate gradients) and the value of the correlation length N_s. Note that Fig. 4.7 only presents the relationship between the NPV and the number of simulation runs obtained from one optimization with one initial random seed but the results of Fig. 4.7 are consistent with those of Table 4.2 which are based on the average of results from five optimizations with five different initial random seeds. From the results of Fig. 4.7 and Table 4.2, we can see that the results obtained from the SG and EnOpt algorithms are very similar. The final NPV obtained from the G-SPSA algorithm is better than those of the other algorithms, but again, this occurs because G-SPSA performs more reservoir simulation runs than the other two algorithms do. For the EnOpt and SG algorithms, the higher the correlation length is, the lower NPV obtained from these algorithms is. The performance of the SmG-SPSA algorithm is worse than that of the other algorithms.

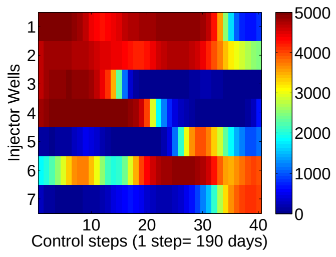
The final remaining oil saturation distributions after optimization are shown in Fig. 4.8. There is little difference between the results obtained from the different algorithms. The final controls for producers and injectors obtained by different algorithms are shown in Figs 4.10 and 4.9. Although the final NPV and the remaining oil distribution obtained by different algorithms are not very different, the estimates of the optimal controls (Figs 4.10 and 4.9) obtained by G-SPSA are substantially different from those obtained from Sm-GSPSA, SG and EnOpt because the last three algorithms used a search direction which is approximately equal to $C_U^2 \nabla J(u)$ which is expected to yield less smooth controls as long is the correlation length of the covariance function on which C_U is based has a correlation length greater than or equal to the length of two control steps. This suggests that sometime there are multiple sets of controls which can achieve essentially the same maximum value of NPV. For the G-SPSA algorithm, almost all final control variables are close to either the lower bound or the upper bound, especially, when the value of correlation length is less than the full value. Note that SmG-SPSA, EnOpt and SG result in the smoother final controls than those obtained from G-SPSA algorithm. For all algorithms, when the correlation length increases, we obtain a smoother final control variable.



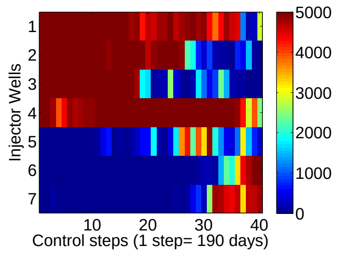
(a) G-SPSA($N_s = 10$)



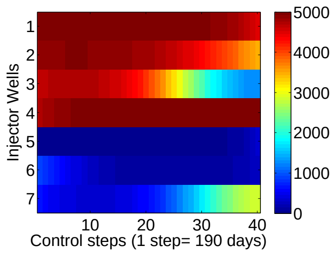
(b) EnOpt($N_s = 10$)



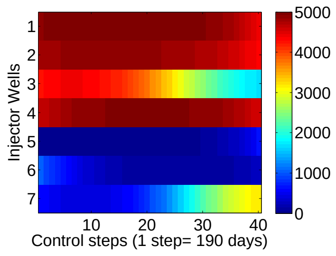
(c) SGD($N_s = 10$)



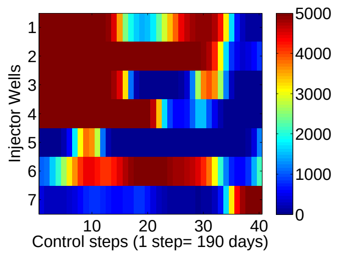
(d) G-SPSA($N_s = 40$)



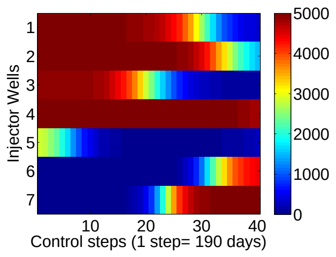
(e) EnOpt($N_s = 40$)



(f) SGD($N_s = 40$)



(g) SmG-SPSA($N_s = 10$)

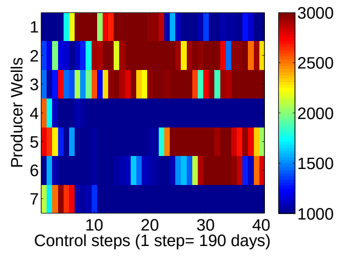


(h) SmG-SPSA($N_s = 40$)

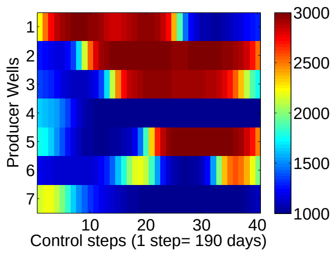
Figure 4.9: The estimated optimal water injection rate well controls obtained from one optimization run of each algorithm ($M = 5$, $\sigma^2 = (1, 1)$; SPSA parameters: $k_{\max} = 280$, $a_0 = 2.0$, $c = 0.1$; EnOpt parameters: $a_0 = 1.0$; SG parameters: $a_0 = 1.0$).

Finally, after working with all algorithms, we can draw some following conclusions based on experimental computations.

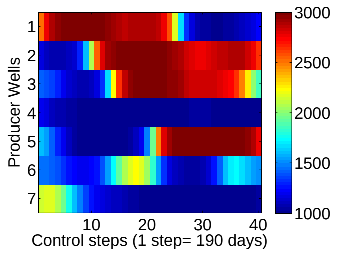
- For small values of N_s, the G-SPSA algorithm always gives a better NPV than the EnOpt and SG algorithms but requires for more reservoir simulation runs to obtain this result. The SmG-SPSA algorithm gives the worst results. As in the previous example, a correlation length equal to the number of control steps is used, the NPV obtained from the EnOpt and SG algorithms is always less



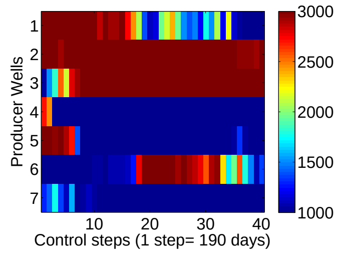
(a) G-SPSA($N_s = 10$)



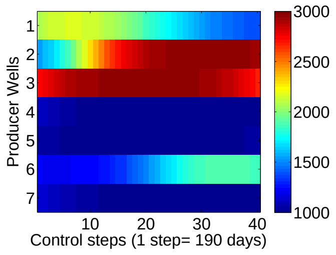
(b) EnOpt($N_s = 10$)



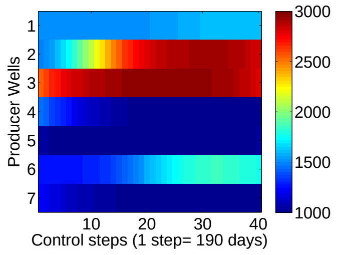
(c) SGD($N_s = 10$)



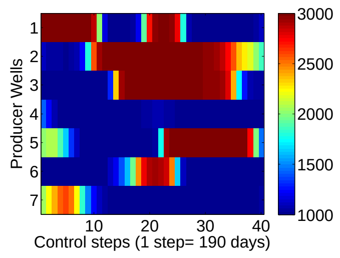
(d) G-SPSA($N_s = 40$)



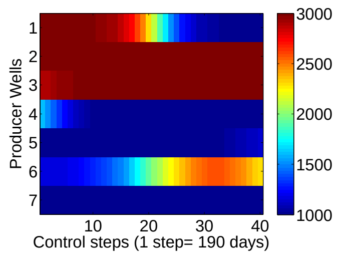
(e) EnOpt($N_s = 40$)



(f) SGD($N_s = 40$)



(g) SmG-SPSA($N_s = 10$)



(h) SmG-SPSA($N_s = 40$)

Figure 4.10: The estimated optimal BPH controls obtained from one optimization run of each algorithm ($M = 5$, $\sigma^2 = (1, 1)$; SPSA parameters: $k_{\max} = 280$, $a_0 = 2.0$, $c = 0.1$; EnOpt parameters: $a_0 = 1.0$; SG parameters: $a_0 = 1.0$).

than that obtained using a small correlation length.

- The estimated optimal NPV and number of simulation runs used with the EnOpt algorithm is very similar to those obtained with the SG algorithm.

- Although the final NPV and the remaining oil distribution obtained from different algorithms are not very different, the final control variables are significantly different. This suggests that sometime there are multiple sets of controls which can achieve essentially the same maximum value of NPV.

Table 4.2: The performance of all algorithms The performance of all algorithms obtained from five optimization runs of each algorithm with five different initial seeds when we change the values of M and N_s ($\sigma^2 = (1, 1)$; SPSA parameters: $k_{\max} = 280$, $a_0 = 2.0$, $c = 0.1$; EnOpt parameters: $a_0 = 1.0$; SG parameters: $a_0 = 1.0$).

Algorithm	$N_s = 10$		$N_s = 40$	
	Simulation	Final NPV	Simulation	Final NPV
G-SPSA($M = 5$)	1680	4.405×10^9	1680	4.398×10^9
SmG-SPSA($M = 5$)	1680	4.398×10^9	1680	4.391×10^9
EnOpt($M = 5$)	662	4.384×10^9	571	4.366×10^9
SG($M = 5$)	776	4.397×10^9	608	4.361×10^9
G-SPSA($M = 10$)	1680	4.418×10^9	1680	4.397×10^9
SmG-SPSA($M = 10$)	1680	4.357×10^9	1680	4.361×10^9
EnOpt($M = 10$)	672	4.394×10^9	588	4.359×10^9
SG($M = 10$)	694	4.387×10^9	570	4.361×10^9

4.4.3 Optimization with simple bound constraints, Brugge case

In this example, we keep the same data and model for the Brugge case used in the previous chapters. We consider only the production optimization step of the closed-loop reservoir management problem. There are 30 vertical wells in the Brugge reservoir including 20 smart production wells and 10 smart water injection wells. Each well has multiple segments that can be controlled individually.

We optimize the well controls for years 10 through 30 based on the mean model obtained by Chen et al. (2010), who used the ensemble Kalman filter with covariance localization to assimilate production data for the first ten years of the reservoir life. This example is identical to the production optimization problem for year 10-30 with an adjoint-gradient method which is reported in Chen et al. (2010).

The production period of the reservoir is divided into 40 control steps, i.e., each control step is 182.5 day. There are 84 control variables for each control step. These control variables are the liquid production rate at each individual segment of the production wells and the water injection rate at each individual segment of the injection wells. The total number of control variables is $40 \times 84 = 3360$. The maximum liquid production rate of each producer segment is 3000 STB/D and, the maximum injection rate of each injector segment is 4000 STB/D. As we previously explained, the control variables are transformed into the log-transform domain in order to eliminate the bound constraints. The minimum value for the rate of each segment in a production or an injection well is 0 STB/D. The minimum BHP constraint for a producer segment and the maximum BHP constraint for an injection well segment, respectively, are 725 psi and 2611 psi. The BHP nonlinear constraints are considered reactively by inputting them directly into the simulator data file. The oil price is $r_{\mathrm{o}} = \$80.0/\mathrm{STB}$ and both the water production and the injection costs are $r_{\mathrm{w}} = r_{\mathrm{winj}} = \$5.0/\mathrm{STB}$. The annual discount rate is 0.1. The initial value for the injection rate of each injection well segment is 1333.3 STB/D and the initial value for the liquid production rate of each production well segment is 700 STB/D.

For the G-SPSA and SmG-SPSA algorithms, we use the steepest-ascent method where the stochastic gradient is approximated by the average of 10 G-SPSA gradients (Eq. 2.17). The controls of each perturbation are sampled from a Gaussian distribution with variance $\sigma^2 = (1, 1)$, of which the first number corresponds to the transformed variable of liquid production rate and the second number corresponds to the transformed variable of water injection rate, and three correlation lengths of $N_s = 20$, $N_s = 30$ and $N_s = 40$. We set the maximum number of allowable iterations equal to 1120 which is about 1/3 the total number of control variables (3360). We obtain the value of $A = 112$. Based on the previous experimental results, we choose the initial step size $a_0 = 2.0$, then we obtain the value of $a = 35$. We set the value of $c_{\min} = 0.06187$ which gives value of $c = 0.1$. The condition for terminating the G-SPSA and SmG-SPSA algorithms is based on the maximum number of allowable iterations. Note that when 10 G-SPSA gradients are used to calculate the average gradient, 112 iterations correspond to 1232 reservoir simulation runs.

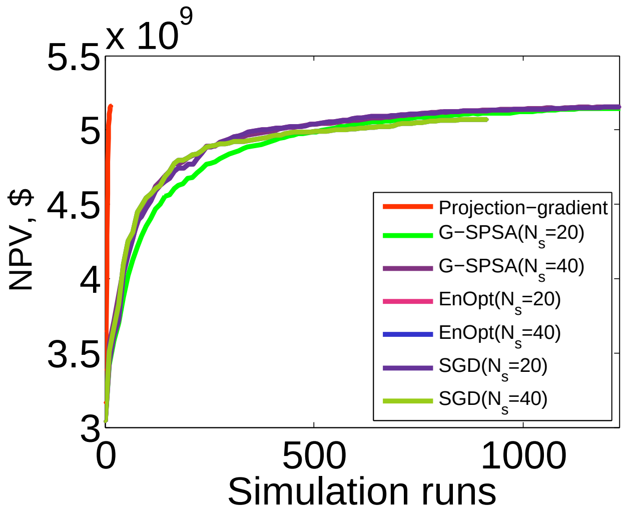
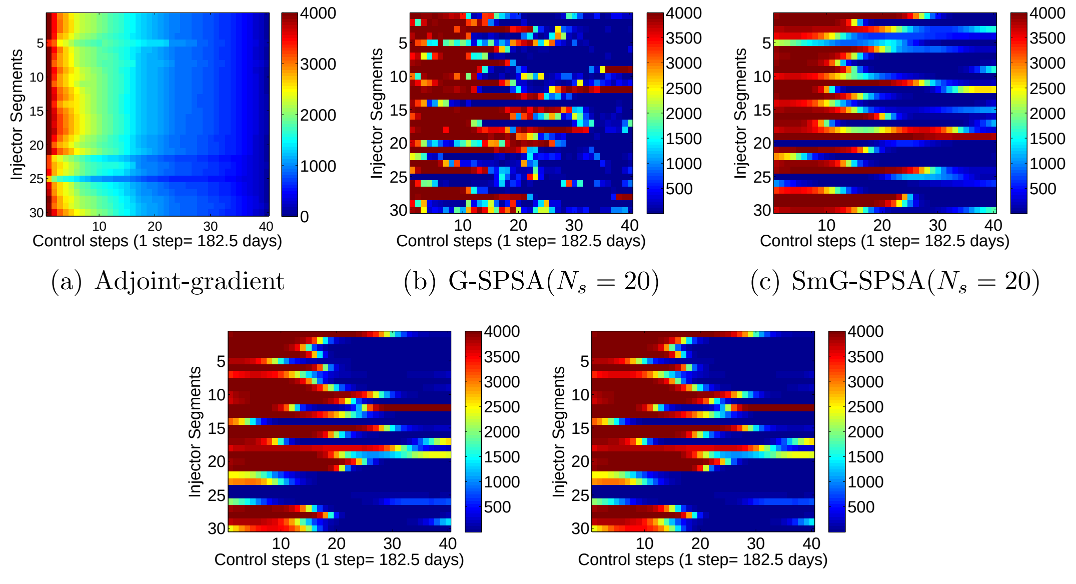


Figure 4.11: NPV versus the number of simulation runs ($M = 10$, $\sigma^2 = (1, 1)$; SPSA parameters: $k_{\max} = 112$, $a_0 = 2.0$, $c = 0.1$; EnOpt parameter: $a_0 = 2.0$; SG parameter: $a_0 = 2.0$).

For the EnOpt and SG algorithms, we use the ensemble size $N_e = 10$ to generate an approximate gradient. The variance $\sigma^2 = (1, 1)$, of which the first number corresponds to the transformed variable of liquid production rate and the second number corresponds to the transformed variable of water injection rate, and three correlation lengths of $N_s = 20$, $N_s = 30$ and $N_s = 40$ are used to construct the covariance C_U. The initial step size is equal to 2.0 and the normalized search directions in Eqs. 4.11 and 4.22 are used to update respectively the transformed variables for the EnOpt algorithm and the SG algorithm, respectively. The condition for terminating EnOpt and SG, respectively, is based on discussion on Section 4.1 and Section 4.2.

For the adjoint-gradient based algorithm (Chen et al., 2010), we set the initial trust-region radius equal to 32 and the upper bound of the trust-region radius equal to 512. The NPV obtained with the adjoint-gradient algorithm is about $\$5.162 \times 10^9$ after 30 simulation runs. For the adjoint-gradient algorithm, we need one forward simulation to calculate NPV and one backward adjoint run to calculate the gradient.



(a) Adjoint-gradient (b) G-SPSA($N_s = 20$) (c) SmG-SPSA($N_s = 20$)

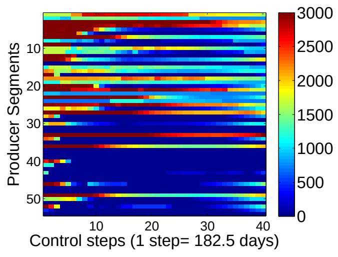
(d) EnOpt($N_s = 20$) (e) SGD($N_s = 20$)

Figure 4.12: The estimated optimal injection well controls obtained from one optimization run of each algorithm ($M = 10$, $\sigma^2 = (1, 1)$; SPSA parameters: $k_{\max} = 112$, $a_0 = 2.0$, $c = 0.1$; EnOpt parameters: $a_0 = 2.0$; SG parameters: $a_0 = 2.0$).

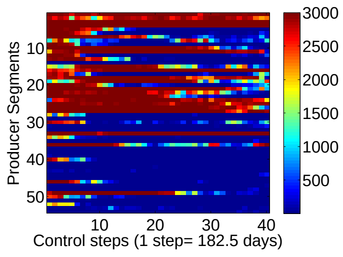
The 30 simulation runs required for the method based on an adjoint gradient consists of 15 forward reservoir simulation runs and 15 adjoint solutions for calculating the gradient of NPV.

The performance of the derivative-free algorithms are summarized in Table 4.3 and Fig. 4.11. In this example, the G-SPSA algorithm using the correlation length of $N_s = 20$ gives almost exactly the same NPV obtained with the SmG-SPSA, SG and EnOpt algorithms after about 1230 reservoir simulation runs. However, when $N_s = 30$ and $N_s = 40$, the NPV's obtained from G-SPSA with those correlation lengths are better than those obtained from the other algorithms using the corresponding correlation length.

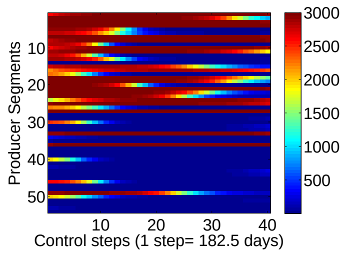
The G-SPSA algorithm converged to the final NPV of \$$5.154 \times 10^9$ which is slightly smaller than the NPV obtained with adjoint-gradient based algorithm. The final controls for the producers and injectors obtained by the derivative-free



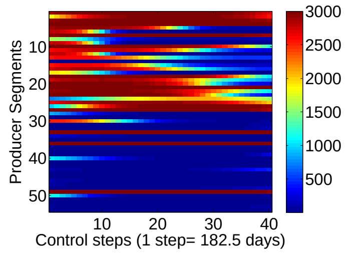
(a) Adjoint-gradient



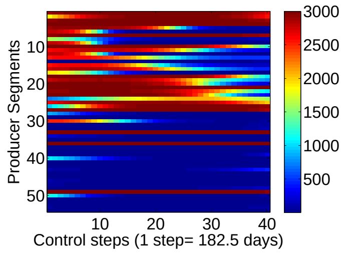
(b) G-SPSA($N_s = 40$)



(c) SmG-SPSA($N_s = 40$)



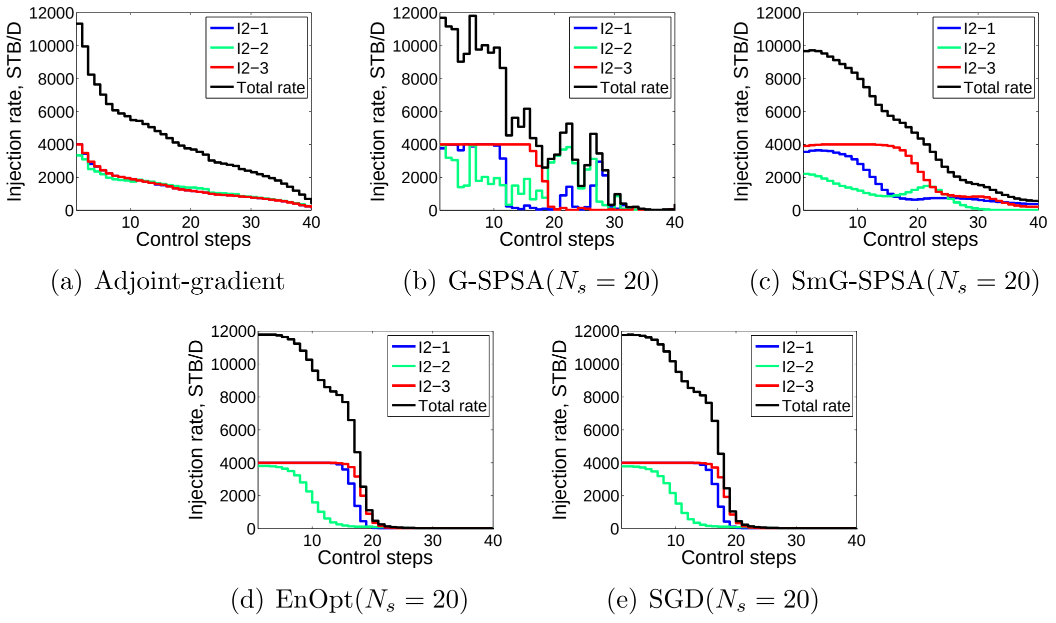
(d) EnOpt($N_s = 40$)



(e) SGD($N_s = 40$)

Figure 4.13: The estimated optimal production well controls obtained from one optimization run of each algorithm ($M = 10$, $\sigma^2 = (1, 1)$; SPSA parameters: $k_{\max} = 112$, $a_0 = 2.0$, $c = 0.1$; EnOpt parameters: $a_0 = 2.0$; SG parameters: $a_0 = 2.0$).

algorithms and the adjoint-gradient based algorithm are shown in Figs. 4.12 and 4.13. Again, the results of Figs. 4.12 and 4.13 show that the final controls obtained by SmG-SPSA, EnOpt and SG are smoother than the final controls obtained from G-SPSA algorithm as expected. Note that not only are the final NPV's obtained by the EnOpt and SG algorithms are very different (Table 4.3), but the estimates of the optimal controls obtained from these two algorithms (Figs. 4.12 and 4.13) are also very similar. This suggests that sometimes there are multiple sets of controls which can achieve essentially the same maximum value of NPV. The controls in Fig. 4.13 show high liquid production rates for the first 30 producing segments, which correspond to the first ten producers. These producers are located at the edge of the main fault and are farther from the injectors than are the other producers. The wells that are close to the injectors (the segments after 30) produce at much lower liquid rates. The estimates of the optimal total water injection rate and segment rates

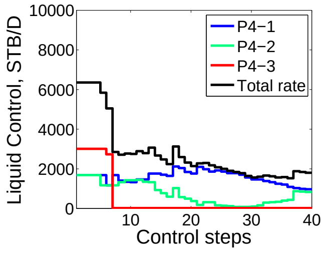


(a) Adjoint-gradient (b) G-SPSA($N_s = 20$) (c) SmG-SPSA($N_s = 20$)

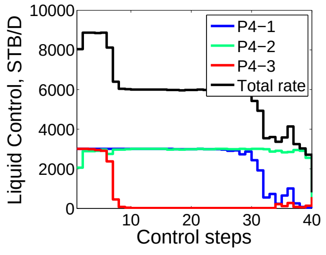
(d) EnOpt($N_s = 20$) (e) SGD($N_s = 20$)

Figure 4.14: The estimated optimal well segment injection rates for Injector 2 obtained from one optimization run of each algorithm ($M = 10$, $\sigma^2 = (1, 1)$; SPSA parameters: $k_{\max} = 112$, $a_0 = 2.0$, $c = 0.1$; EnOpt parameters: $a_0 = 2.0$; SG parameters: $a_0 = 2.0$).

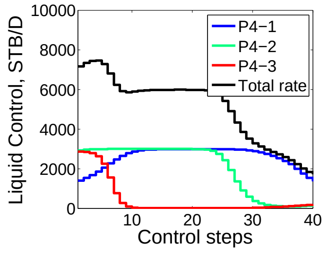
for Injector 2 as a function of time are shown in Fig. 4.14. We see that injection rate controls estimated from G-SPSA are not as smooth as those estimated from the other algorithms. Fig. 4.15 shows the optimized rate controls for the three segments of producer P4 from all the algorithms.



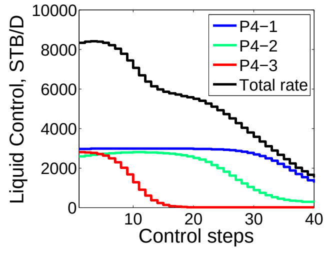
(a) Adjoint-gradient



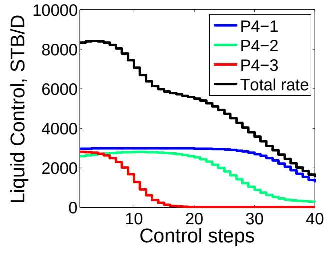
(b) G-SPSA($N_s = 40$)



(c) SmG-SPSA($N_s = 40$)



(d) EnOpt($N_s = 40$)



(e) SGD($N_s = 40$)

Figure 4.15: The estimated optimal well segment liquid production rates for Producer 4 obtained from one optimization run of each algorithm ($M = 10$, $\sigma^2 = (1, 1)$; SPSA parameters: $k_{\max} = 112$, $a_0 = 2.0$, $c = 0.1$; EnOpt parameters: $a_0 = 2.0$; SG parameters: $a_0 = 2.0$).

Table 4.3: The performance of algorithms obtained from one optimization run of each algorithm when we change the values of correlation length ($M = 10$, $\sigma^2 = (1, 1)$; SPSA parameters: $k_{\max} = 112$, $a_0 = 2.0$, $c = 0.1$; EnOpt parameters: $a_0 = 2.0$; SG parameters: $a_0 = 2.0$).

Algorithm	$N_s = 20$		$N_s = 30$		$N_s = 40$	
	Simul-	Final NPV	Simul-	Final NPV	Simul-	Final NPV
G-SPSA	1232	5.142×10^9	1232	5.154×10^9	1232	5.153×10^9
SmG-SPSA	1232	5.140×10^9	1232	5.112×10^9	1232	5.093×10^9
EnOpt	1230	5.153×10^9	664	5.063×10^9	914	5.071×10^9
SGD	1230	5.152×10^9	664	5.063×10^9	914	5.071×10^9

CHAPTER 5

DISCUSSION AND CONCLUSIONS

The main contribution of this work was the development and implementation of derivative-free optimization methods for large-scale production optimization problems with bound, linear and nonlinear constraints. Here, we use the terminology derivative-free rather loosely to return to methods that do not employ an adjoint solution to calculate the gradient of the objective function. In reality, our algorithm used approximation to the gradient or preconditioned gradient that can be calculated far more efficiently than the finite-different approximation of each derivative in the gradient. In the production optimization literature, well controls are often adjusted using a gradient-based algorithm (Brouwer and Jansen, 2004; Jansen et al., 2005; Sarma et al., 2005, 2008; Chen et al., 2010) with the gradient of the objective function that we wish to maximize computed by the adjoint method using an adjoint formulation similar to the one presented by Li et al. (2003) for history matching. Unfortunately, commercial simulators have limited capability for computing the gradients needed for general constrained optimization with the adjoint method, which provides the motivation for this work. In this work, we developed and applied computationally efficient modification of the SPSA algorithm for constrained production optimization problems. We also investigated combined direction stochastic approximation algorithms, including a novel one we developed, but the results obtained from these algorithms never proved superior to the modified SPSA algorithm. Via computational experiments we developed guidelines for choosing the parameters in the SPSA algorithm. The modified SPSA algorithm which uses Gaussian perturbations to approximate a stochastic gradient was compared to the performance of an SPSA algorithm which uses sample from the ± 1 Bernoulli distribution. The computational results clearly indicate that a more efficient and robust algorithm is obtained by using Gaussian perturbations. We also compared the results of the modified SPSA algorithm which uses Gaussian perturbations with an ensemble-based optimization algorithm and an algorithm which uses a preconditioned simplex gradient. The performance of the three algorithms were found to be similar and in Chapter 4, we established a theoretical connection between the methods. In fact, we showed that all algorithms generate an approximation of the same preconditioned gradients assuming Gaussian perturbations are used in all three algorithms. Although the modified SPSA algorithm was originally developed for the unconstrained optimization problems, we have shown that it can be applied in the inner-loop of the augmented Lagrangian method for general nonlinear constraints as long as we first remove the bound constraint by using a log-transformation of variables. Worked in the log-transform domain, SPSA with Gaussian perturbations is used to estimate the gradient of the augmented Lagrangian function at each inner-loop iteration. This augmented Lagrangian algorithm was applied successfully to the well known Brugge test case and was able to achieve an optimal NPV almost identical to one based on an algorithm which used an accurate gradient computed with an adjoint method, however, the adjoint-based algorithm was far more computationally efficient than the SPSA-based procedure.

Based on the extensive computations we have done, we can reasonably state that for production optimization problems of interest, the following conclusions are valid

- The one-sided simultaneous perturbation with both Bernoulli distribution and Gaussian distribution results in a more computationally efficient algorithm than is obtained with a two-sided simultaneous perturbation;

- The performance of the SPSA algorithm with a Gaussian perturbation (G-SPSA) is always better than the algorithm's performance with Bernoulli perturbation (B-SPSA);

- The performance of ComG-SPSA1 and ComG-SPSA2 is not better than the G-SPSA algorithm's performance;

- For both Bernoulli perturbation and Gaussian perturbation, the optimal number (M) of SPSA gradients to use to construct the search direction in a steepest-ascent algorithm is on the order of 5 to 10;

- When we use the Gaussian perturbation, a small correlation length results in very rough final well controls (which may not be very practical). Larger correlation lengths give smoother final control values. Based on our experimentations, a correlation length on the order of one half the number of the control steps should be reasonable for a large class of problem.

- We can say that a good value of the implicit SPSA parameter $c_{\min}$ is from 0.01 to 1.0 for the bound constrained problems when we work on the log-transform domain;

- In a production optimization problem, the maximum allowable iterations should be roughly equal to the total number of control variables, if sufficient computation time is available. But when the number of control variables is large, setting the maximum number of allowable iterations equal to 1/3 to 1/2 the number of control variables will still achieve an acceptable net present value (NPV). An initial stepsize in the range from 1.0 to 2.5 gave good results in all examples.

- When bound constraints are removed by applying log-transformation, it is best to use the initial guess for each control variable (in the original domain)

$u_i \approx (u_i^{\text{up}} + u_i^{\text{low}})/2$, where u_i^{up} and u_i^{low}, respectively, represent the upper bound and lower bound on u_i. Using u_i^{up} or u_i^{low} is often a poor choice as it makes difficult to escape the from the bound.

- Recall that N_s is equal to the number of perturbations used to estimate the preconditioned gradient. For of N_s on the order of 5 to 10, the G-SPSA algorithm always gives a better NPV than the EnOpt and SG algorithms do. However, the total number of simulation runs of this G-SPSA algorithm are always higher than those of other algorithms because SPSA continues until the pre-specified number of iterations is reached.

- The performance of the EnOpt algorithm and the performance of the SG algorithm are very similar which is expected based on the theoretical results established in chapter 4.

- Although the final NPV and the remaining oil distribution obtained from different algorithms are not very different, the final control variables are significantly different. This suggests that sometimes there are multiple sets of controls which can achieve essentially the same maximum value of NPV.

- In the Brugge case, the best NPV obtained in this experiment is the same as the NPV obtained with the adjoint-gradient based algorithm (Chen et al., 2011a). This is extremely encouraging result as our objective is to obtain an optimization algorithm which does not require an adjoint code to computer the gradient; For the Brugge example specifying the maximum number of allowable iterations equal to one-third the total number of control variables gave good results at least compared to the NPV obtained using an accurate gradient (Chen et al., 2011a).

BIBLIOGRAPHY

Bangerth, W., H. Klie, M. Wheeler, P. Stoffa, and M. Sen, On optimization algorithm for the reservoir oil well placement problem, *Computational Geosciences*, **10**, 303–319, 2006.

Barker, J. W., M. Cuypers, and L. Holden, Quantifying uncertainty in production forecasts: Another look at the PUNQ-S3 problem, *SPE Journal*, **6**(4), 433–441, 2001.

Bertsekas, D. P., *Nonlinear Programming*, Athena Scientific, Belmont, Massachusetts, 1995.

Bertsekas, D. P. and J. N. Tsitsiklis, Gradient convergence in gradient methods with errors, *SIAM Jounal Optim.*, **10**, 627–642, 2003.

Bortz, D. and C. Kelley, *Computational methods for optimal design and control*, vol. 24, chap. The simplex gradient and noisy optimization problems, pp. 77–90, Birkhauser, 1998.

Brouwer, D. and J. Jansen, Dynamic optimization of water flooding with smart wells using optimial control theory, *SPE Journal*, **9**(4), 391–402, 2004.

Chen, C., *Adjoint-Gradient-Based Production Optimization With The Augmented Lagrangian Method*, PhD. thesis, University of Tulsa, Tulsa, Oklahoma, USA, 2011.

Chen, C., G. Li, and A. Reynolds, Closed-loop reservoir management on the Brugge test case, *Computational Geosciences*, **14**(4), 691–703, 2010.

Chen, C., G. Li, and A. Reynolds, Robust constrained optimization of short and long-term NPV for closed-loop reservoir management, in *Proccedings of the SPE Reservoir Simulation Symposium, The Woodlands, Texas, USA, 21-23 February*, SPE 141314, 2011a.

Chen, C., G. Li, and A. C. Reynolds, Robust constrained optimization of short and long-term NPV for closed-loop reservoir management,, in *Proceedings of SPE Reservoir Simulation Symposium*, SPE 141314, (to appear in *SPE Journal*), 2011b.

Chen, Y., D. Oliver, and D. Zhang, Efficient ensemble-based closed-loop production optimization, *SPE Journal*, **14**(4), 634–645, 2009.

Conn, A., N. Gould, and P. Toint, *LANCELOT: A Fortran Package for Large-Scale Nonlinear Optimization (Release A)*, Springer-Verlag, New York, 1992.

Conn, A. R., N. Gould, and P. Toint, *Trust-Region Methods*, SIAM, Philadelphia, 2000.

Conn, A. R., ScheinbergKatya, and L. N. Vicente, Geometry of interpolation sets in derivative free optimization, *Mathematical Programming*, **111**(1-2), 141–172, 2006.

Custsódio, A. L. and L. N. Vicente, Using sampling and simplex derivatives in pattern search methods, *SIAM Journal on Optimization*, **18**, 537–555, 2007.

de Montleau, P., A. Cominelli, D. R. K. Neylon, I. Pallister, O. Tesaker, and I. Nygard, Production optimization under constraints using adjoint gradients, in *Proceedings of the 10th European Conference on the Mathematical Oil Recovery - Amsterdam, 4-7 September*, SPE 109805, 2006.

Dehdari, V. and D. Oliver, Sequential quadratic programming SQP for solving constrained production optimization-case study from Brugge field, in *Proccedings of* *the SPE Reservoir Simulation Symposium, The Woodlands, Texas, USA, 21-23 February*, SPE 141589, 2011.

Fabian, V., On asymptotic normality in stochastic approximation, *The Annals of Mathematical Statistics*, **39**, 1327–1332, 1968.

Floris, F. J. T., M. D. Bush, M. Cuypers, F. Roggero, and A.-R. Syversveen, Methods for quantifying the uncertainty of production forecasts: A comparative study, *Petroleum Geoscience*, **7**(SUPP), 87–96, 2001.

Forouzanfar, F., G. Li, and A. Reynolds, A two-stage well placement optimization method based on adjoint gradient, in *Proceedings of the SPE Annual Technical Conference and Exhibition*, SPE 135304, 2010.

Gao, G., G. Li, and A. C. Reynolds, A stochastic algorithm for automatic history matching, *SPE Journal*, **12**(2), 196–208, 2007.

Gao, G., M. Zafari, and A. C. Reynolds, Quantifying uncertainty for the PUNQ-S3 problem in a Bayesian setting with RML and EnKF, *SPE Journal*, **11**(4), 506–515, 2006.

Jansen, J., D. Brouwer, G. Naevdal, and C. van Kruijsdijk, Closed-loop reservoir management, *First Break*, **23**, 43–48, 2005.

Kelley, C., Detection and remediation of stagnation in Nelder-Mead algorithm using a sufficient decrease condition, *SIAM Journal on Optimization*, **10**, 43–55, 1999.

Li, G. and A. Reynolds, Uncertainty quantification of reservoir performance predictions using a modification of SPSA algorithm, *Computational Geosciences*, **10(3)**, 451–462, 2011.

Li, R., A. C. Reynolds, and D. S. Oliver, History matching of three-phase flow production data, *SPE Journal*, **8**(4), 328–340, 2003.

Lorentzen, R., A. Berg, G. Naevdal, and E. Vefring, A new approach for dynamic optimization of water ooding problems, in *the SPE Intelligent Energy Conference and Exhibition*, SPE 99690, 2006.

Luenberger, D. G., *Linear and Nonlinear Programming*, Addison-Wesley Publishing Company, 1984.

Nocedal, J. and S. J. Wright, *Numerical Optimization*, Springer, New York, 1999.

Nwaozo, J., *Dynamic Optimization of a Water Flood Reservoir*, Master's thesis, University of Oklahoma, Norman, Oklahoma, 2006.

Peters, E., R. Arts, G. Brouwer, and C. Geel, Results of the brugge benchmark study for flooding optimisation and history matching, in *Proceedings of the SPE Reservoir Simulation Symposium, 2-4 February*, SPE 119094, 2009.

Peters, L., R. Arts, G. Brouwer, C. Geel, S. Cullick, R. Lorentzen, Y. Chen, K. Dunlop, F. Vossepoel, R. Xu, P. Sarma, A. Alhuthali, and A. Reynolds, Results of the Brugge benchmark study for flooding optimisation and history matching, *SPE Reservoir Evaluation & Engineering*, **13**(3), 391–405, 2010.

Powell, M., The BOBYQA algorithm for bound constrained optimization without derivatives, DAMTP 2009/NA06, Centre for Mathematical Sciences, University of Cambridge, 2009.

Powell, M. J., *The NEWUOA software for unconstrained optimization without derivatives* ***in*** *large-scale nonlinear optimization, eds G. Di Phillo and M. Roma*, Springer, 2006.

Rao, C. R., *Linear Statistical Inference and Its Applications*, John Wiley & Sons, New York, 1965.

Reynolds, A. C., M. Zafari, and G. Li, Iterative forms of the ensemble Kalman filter, *Proceedings of 10th European Conference on the Mathematics of Oil Recovery*, 2006.

Sarma, P., W. Chen, L. Durlofsky, and K. Aziz, Production optimization with adjoint models under nonlinear control-state path inequality constraints, *SPE Reservoir Evaluation & Engineering*, **11**(2), 326–339, 2008.

Sarma, P., L. Durlofsky, and K. Aziz, Implementation of adjoint solution for optimal control of smart wells, in *Proceedings of the SPE Reservoir Simulation Symposium*, SPE 92864, 2005.

Schraudolh, N. N. and T. Graepel, Combining conjuagate direction methods with stochastic approximation of gradient, *Proceeding of the Ninth International Workshop on Artificial Intelligence and Statistic*, pp. 7–13, 2002.

Shuai, Y., C. White, H. Zhang, and T. Sun, Using multiscale regularization to obtain realistic optimal control stratergies, in *Proccedings of the SPE Reservoir Simulation Symposium, The Woodlands, Texas, USA, 21-23 February*, SPE 142043, 2011.

Spall, J. C., Multivariate stochastic approximation using a simulataneous perturbation gradient approximation, *IEEE Transactions Automat. Control.*, **37**(3), 332–341, 1992.

Spall, J. C., Implementation of the simultaneous perturbation algorithm for stochastic optimization, *IEEE Transactions on Aerospace and Electronic Systems*, **34**(3), 817–823, 1998.

Spall, J. C., *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*, Wiley, Hoboken, NJ, 2003.

Sudaryanto, B. and Y. C. Yortsos, Optimization of fluid front dynamics in porous media using rate control. I. equal mobility fluids, *Physics of Fluids*, **12**(7), 1656–1670, 2000.

Tseng, P., Fortified-descent simplicial search method: A general approach, *SIAM Journal on Optimization*, **10**, 269–288, 1999.

van Essen, G., M. Zandvliet, P. V. den Hof, O. Bosgra, and J. Jansen, Robust waterflooding optimization of multiple geological scenarios, *SPE Journal*, **14**(1), 202–210, 2009.

Wang, C., G. Li, and A. C. Reynolds, Production optimization in closed-loop reservoir management, *SPE Journal*, **14**(3), 506–523, 2009.

Wang, I.-J. and J. C. Spall, Stochastic optimisation with inequality constraints using simultaneous perturbations and penalty functions, *International Journal of Control*, **81**(8), 1232–1238, 2008.

Xu, Z., A combined direction stochastic approximation algorithm, *Optimization Letters*, **4**(1), 117–129, 2010.

Zakirov, I. S., S. I. Aanonsen, E. S. Zakirov, and B. M. Palatnik, Optimizating reservoir performance by automatic allocation of well rates, in *Proceedings of the 5th European Conference on the Mathematical Oil Recovery - Leoben, Austria, 3-5 September*, 1996.

Zandvliet, M., O. Bosgra, J. Jasen, P. V. den Hof, and J. Kraaijevvanger, Bang-bang control in reservoir flooding, in *Proceedings of the 10th European Conference on the Mathematical Oil Recovery - Amsterdam, 4-7 September*, 2006.

Zandvliet, M., O. Bosgra, J. Jasen, P. V. den Hof, and J. Kraaijevvanger, Bang-bang control and sigular arcs in reservoir flooding, *Journal of Petroleum Science and Engineering*, **58**, 186–200, 2007.

Zhang, K., G. Li, A. C. Reynolds, L. Zhang, and J. Yao, Optimal well placement using and adjoint gradient, *Journal of Petroleum Science and Engineering*, **73**, 220–226, 2010.

Zhao, H., C. Chen, S. Do, G. Li, and A. Reynolds, Maximization of a dynamic quadratic interpolation model for production optimization, in *Proccedings of the SPE Reservoir Simulation Symposium, The Woodlands, Texas, USA, 21-23 February*, SPE 141317, 2011.