AES/PE/13-44 **Constraint Handling in Life-cycle Optimization Using Ensemble Gradients**

December 18, 2013 **Marcel Alim**

Title	:	Constraint Handling in Life-cycle Optimization Using Ensemble Gradients

Author(s)	:	Marcel Alim

Date	:	December 18, 2013
Professor(s)	:	Jan Dirk Jansen
Supervisor(s)	:	Olwijn Leeuwenburgh (TNO)
		Paul Egberts (TNO)
TA Report number	:	AES/PE/13-44

Postal Address	:	Section for Petroleum Engineering
		Department of Geoscience & Engineering
		Delft University of Technology
		P.O. Box 5028
		The Netherlands
Telephone	:	(31) 15 2781328 (secretary)
Telefax	:	(31) 15 2781189

Acknowledgements

I would never have been able to finish my master thesis without the guidance from my supervisors, help from my friends, and support from my family. I would like to express my deepest gratitude to my supervisors Professor Jan Dirk Jansen, Olwijn Leeuwenburgh, Paul Egberts, and Rahul Fonseca for their supervision, support, guidance, advice, and patience for my research in constrained optimization. They patiently corrected and revised my writing and also helped me adapting in a new working environment. I feel really lucky to get this opportunity, working on optimization problem together with the best minds in the discipline while also improving my programming skills. My words alone will never be enough to show how much I grateful for all the things they have done to help me. I would also say thank you to BP and especially to Astor-Lonice Bal for financially supporting my study for two years. Without the scholarship, I would never be in here in the first place.

I would like to thank Danny, Iwan, Ade, and Allan for being good friends and my partners in crime for the amazing two years I have here. Many thanks to Matei, Benjamin, Sebastian, Linde, and other interns and people at TNO Utrecht; it would have been a lonely office without you. Special thanks to Trang for all your kindness, support, patience, and especially your cooking to keep me well feed. Michelle for helping me with my presentation, I owe you big time. Lesly for all the fun time, Tania for bringing me to Delft, and Tante Jean and Om Stanley for their caring and attention, I feel like I have a family in here. I would also say thank you to all my professors, friends, and colleagues (you know who you are) whom have made my life in the Netherlands so colourful and memorable.

Most importantly, how I am blessed and grateful for my family. Even though you are far away, I can always feel your love, support, blessing, and prayer for me to finish my study. Thank you Mom and Dad for letting me study in here and supporting my life path. Thank you my lovely sister Karina for growing up with me and listening to my stories. And lastly, thank you God. Without You, nothing else matters.

Delft, 22 October 2013

Marcel

Abstract

Constrained optimization is the process of optimizing an objective function with respect to some variables in the presence of constraints on those variables themselves or on some function of those variables. This thesis focused on using the Ensemble Optimization method to improve the NPV (Net Present Value) as the objective function of waterflooding a reservoir with an L-shaped sealing fault under constraints. The optimization controls are injection rates for the input-constrained optimization and valves opening for the output-constrained optimization. The constraints are field injection rate for the input-constrained optimization and field production rate for the output-constrained optimization. Three Matlab optimization methods were tested, of which the SQP (Sequential Quadratic Programming) method performed the best. For dealing with the constraints, it is better to let the optimizer handle them instead of the simulator. Two ways to help the optimizer to have a better constraint adherence are by using the constraint scaling and improving the quality of the gradients. Having too many variables may lead to a lower objective function due to the approximate gradients' inaccuracies. Regularization (smoothing) can help to improve the objective function in this problem.

Table of Content

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Closed-loop Life-cycle Reservoir Management

The concept of closed-loop or real-time reservoir management and life-cycle optimization has been described in different forms before such as *e-fields* (Litvak et al. 2002), *smart fields* (Kapteijn and Muessig, 2003; Potters and Kapteijn, 2005), *self-learning reservoir management* (Saputelli et al. 2005), and many more. The general idea of closed-loop reservoir management is to use control and optimization concepts from process control, and data assimilation methods from oceanography and meteorology to find a reservoir management strategy that maximizes a specified economic objective (Net Present Value or NPV) or oil production. (Brouwer et al. (2004), Jansen et al. (2005, 2008, 2009), Naevdal et al. (2006), Chen (2008)).

Closed-loop reservoir management of two components: model updating and life-cycle optimization. Sometimes a third component is introduced in the form of upscaling or model reduction. The objective of model updating is to minimize the difference between the model and the real reservoir (or production data) available in time because the fluid and rock properties are highly uncertain due to limited access to reservoir information. This model updating is done via data assimilation, also known as history matching, and is either performed on the (dynamic) flow model, or, preferably, on the (static) geological model. By having accurate models, forecasting and operation strategy planning can be improved. The current most popular method to handle a large number of variables is Ensemble Kalman Filter (EnKF) in which a set of models is updated.

Reservoir up/down-scaling, even though it is more common to do upscaling than downscaling, is important to improve the computational efficiency. By using low-order models, the simulation time for optimization or data assimilation can be reduced significantly since usually it is required to do tens to hundreds of system response simulation (Jansen, 2012). Downscaling is commonly used as a way to check whether the upscaling is representative enough. Another reason to use low-order models is that the level of detail in system models should be adapted to the available information and the extent of control (observability, to what extent can a state variable be reconstructed from the outputs, and controllability, to what extent can a state variable be influenced by the inputs) (Jansen, 2012).

Slightly different from short-term production optimization, the goal of life-cycle optimization is to maximize the oil production (or NPV) in the long term while obeying some constraints or contracts. An optimal short-term production strategy does not necessarily conform to an optimal life-cycle strategy. The hypothesis underlying the concept of closed-loop reservoir management is that it will be possible to significantly increase the life-cycle value by changing reservoir management from a periodic to a near-continuous model-based controlled activity (Jansen et al. 2009).

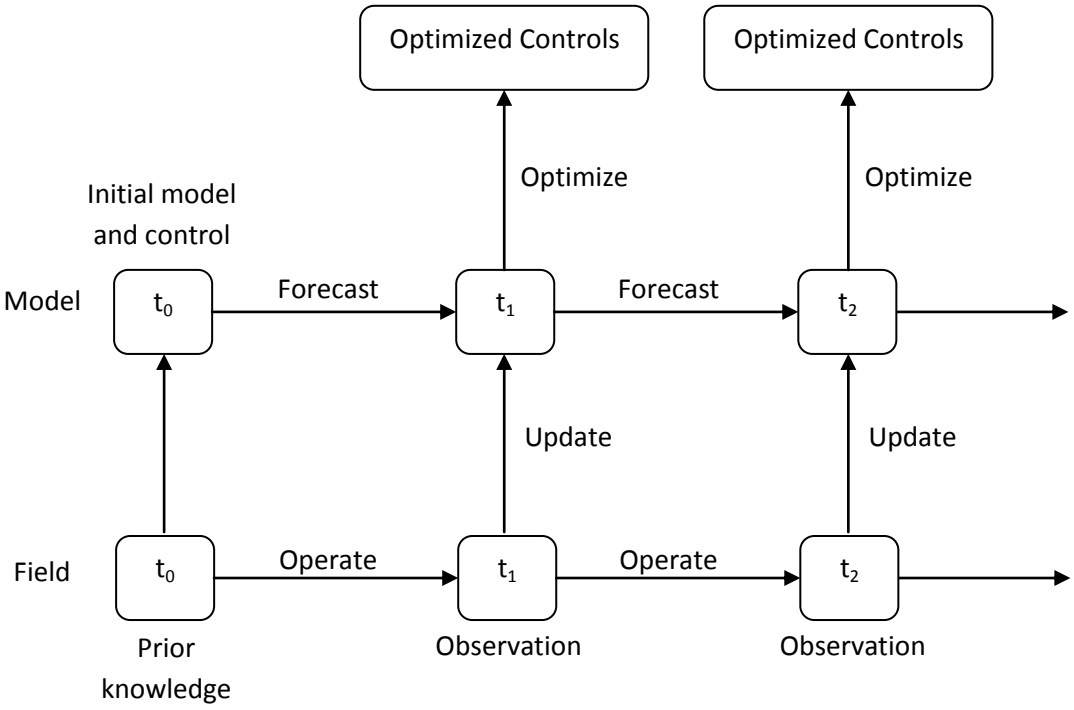


Figure 1. Flowchart for closed-loop optimization

In reservoir management, there are a lot of variables that can be changed or controlled, from injection rates, production rates, injection composition, inflow control valve (ICV) openings in smart wells, and well pressures. All these variables that can be controlled are called *well control variables* or briefly *control variables*, denoted with a vector **u**. In addition, there are also some variables that are not related directly to the wells but still part of reservoir management, for example variables related to well placement and drilling schedule. The number of control variables (i.e. the number of elements in the vector **u**) is proportional to the number of wells and the number of times at which the control variables are changed. There is also another type of variable called *state variable*. State variables are the variables that are fundamental to the reservoir state or condition, and calculated through reservoir simulation. Examples of state variables are water and/or oil saturations, pressure, or component accumulation. Finally, there are output variables, which are (combinations of) state variables that can be observed through measurements. For example, if well pressure is chosen to be the control variable then well rate will typically be the output variable in that well, and vice versa.

1.2 Life-cycle Optimization

The objective of life-cycle optimization is to maximize an objective function over the remaining expected life of a reservoir, while honoring constraints that could result from physical, environmental, or economical requirements. The objective function is usually either the net present value (NPV) or the hydrocarbon recovery. Even though life-cycle optimization theoretically can improve the objective function of a field, in reality most oil companies do not use this life-cycle optimization. Most of them just use trial and error or a simple and pragmatic strategy to do the optimization.

The main reasons why companies do not use formal life-cycle optimization are the difficulty to quantify geological interpretations and the large uncertainty in the reservoir model properties which will reduce the value of optimization process that is based on a single model (Jansen, 2012). To mitigate the uncertainties, one can generate a series of ensemble models based on statistical distributions and use the models to find the optimal strategy to manage the reservoir instead of only working the one most probable model and find the optimal strategy based on that one model (Van Essen et al. 2009).

The optimization that takes into account uncertainty in the reservoir model is called robust optimization while the optimization that only tries to optimize the well control given a single reservoir model is called deterministic optimization (Van Essen et al. 2009, Paul Egberts, TNO notes). In this study, the focus will be on life-cycle optimization, and in particular output constraints handling. Thus data assimilation, reservoir upscaling, and robust optimization will not be discussed any further.

Below is an example of the difference between an optimized strategy (Figure 2, right side) and a constant water injection rate strategy (Figure 2, left side) for a homogeneous reservoir model with an L-shaped sealing fault, taken from SPE paper 105764.

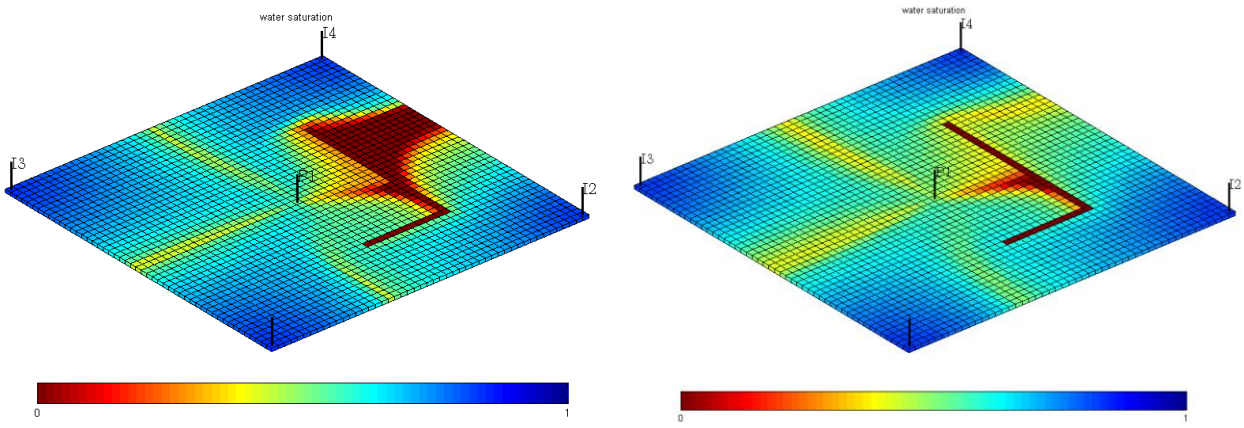


Figure 2. Saturation after waterflood between two different strategies

From Figure 2 above, the difference between the optimized and constant ICV opening strategy is very distinct. In the constant strategy, the injectors will just be injecting water constantly over time while in optimized production strategy, the optimizer will find an optimal schedule to inject water to obtain maximum NPV. The general idea of the optimized strategy in this L-shaped fault reservoir model is that during early times the injector 2 will be widely opened while injector 4 will only be slightly opened to push the oil towards injector 1 and 4. Later on, injector 4 will be widely opened as well to push the accumulated oil near injector 4 to the producer in the middle of the reservoir. By implementing this strategy, higher NPV can be obtained. Below is the difference in control (i.e. the ICV opening) between the constant and optimized ICV strategies.

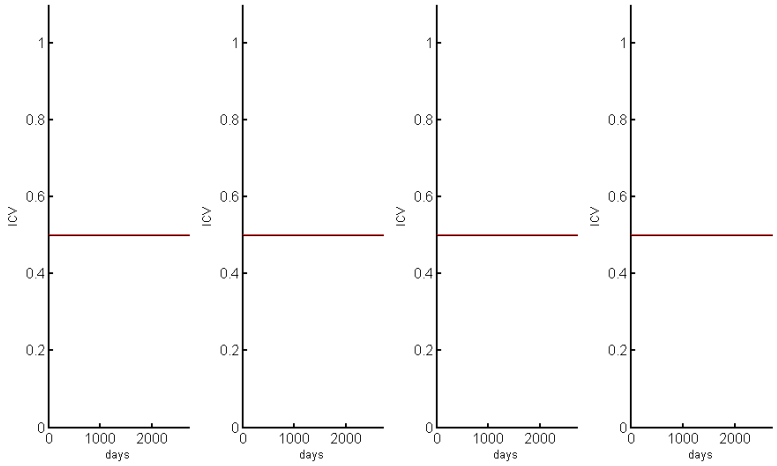


Figure 3. Controls for constant ICV strategy

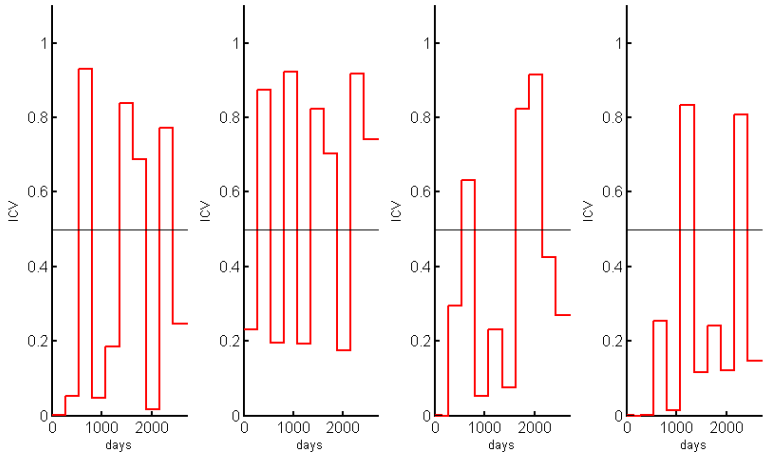


Figure 4. Controls for optimized strategy

There are quite a lot of different optimization methods, ranging from gradient-free methods, gradient-based methods, to sequential methods. Gradient-free methods are computationally expensive for many variables. The basic idea of using gradient information is to use the gradient to

determine the direction in which to update control variables to improve the objective function value. There are two common strategies on how to update the control variables: the line search strategy and the trust region strategy. The idea of the line search strategy is to first determine the direction of the next step, and then determine how far one should move along that direction, i.e. the step length, while for the trust region strategy, the idea is to first determine the step length of the trust region size and then determine the direction one should move along. Both strategies can be used inside an iterative procedure to find the (local) optimum of the objective function.

The trust region strategy defines a "trustworthy" region based on a model (usually a quadratic approximation) of the objective function, and then minimizes the approximated model within this region. If the model is unacceptable, the trust region radius is reduced, and if the model is acceptable, the radius will be the same or even expanded in the next iteration.

The quadratic approximation model $m_k(\mathbf{d})$ of the objective function $J(\tilde{\mathbf{u}}_\mathbf{k} + \mathbf{d})$ at point $\tilde{\mathbf{u}}_\mathbf{k}$ is:

$$m_k(\mathbf{d}) = J_k + \nabla {J_k}^T \mathbf{d} + \frac{1}{2}\mathbf{d}^T \mathbf{B_k}\, \mathbf{d} \tag{1}$$

although it is more common to approximate the model $m_k(\mathbf{d})$ with just linear approximation:

$$m_k(\mathbf{d}) = J_k + \nabla {J_k}^T \mathbf{d} \tag{2}$$

In gradient-based methods, there are two well-known methods to calculate the gradient: the finite difference method and adjoint method. For finite difference method, for n number of control variables it requires $n + 1$ simulation runs to calculate the gradient. This makes finite difference method impractical when dealing with problems with large numbers of control variables, for example life-cycle optimization and history matching. Different than the finite difference method, the adjoint method is a method to calculate the exact gradient of the objective function and only needs two runs regardless the number of optimization variables: a normal forward simulation and subsequently its adjoint in a backward simulation. It is well known that adjoint method is a very efficient method because of this reason. Unfortunately, the implementation of adjoint method requires access to the simulator and extensive implementation efforts.

1.3 Constraint Types in Life-cycle Optimization

As already mentioned before, it is common to have constraints that need to be obeyed during the production due to physical, environmental, or economical restrictions. There are often many constraints that need to be handled in real life-cycle optimization, for example water/gas injection availability and how to distribute them, field (oil or liquid) production rate, and water cut. All these limitations will make the optimization process become more difficult because now all these constraints have to be taken care of while still trying to optimize the objective function.

Some of the constrained life-cycle optimization problems are more difficult than the others. If physical limits on controls are the only constraint that needs to be obeyed, the constraints are called

bound constraints and can be expressed as $a \leq \mathbf{u} \leq b$ where a is the lower boundary, b is the upper boundary, and $\mathbf{u}$ is the control. For example of these bound, a fully opened ICV is valued as 1 and a closed ICV is valued as 0 and therefore all ICV opening values must be between the bounds of 0 and 1.

If a constraint can be expressed explicitly in terms of the controls (i.e. the inputs), then it is called an input constraint. An input constraint is typically a linear equality or inequality constraint in life-cycle optimization problems (e.g. maximum field injection rate when well rates are the controls) and can be expressed as $\mathbf{c}(\mathbf{u}) \leq 0$.

The other type of constraint where the constraints are explicit functions of the output is called output constraint. These outputs themselves are functions of the inputs (i.e. the controls). From mathematical optimization perspective, an optimization problem with output constraints is more difficult than with input constraints because:

- It requires a reservoir simulation to evaluate the output constraints,
- It is not easy to determine (approximate) gradients of the constraints with respect to the control variables,

while for input-constrained optimization problems, the value and an accurate analytic gradient of the constraints can be obtained easily without needing a reservoir simulation.

These output constraints can be expressed as $\mathbf{c}\left(\mathbf{y}\left(\mathbf{x}(\mathbf{u})\right)\right) \leq 0$ where $\mathbf{y}$ is the vector of output variables and $\mathbf{x}$ is the vector of state variables. An example of an output constraint is maximum field liquid production when ICV settings are the controls.

There are several methods that can be used to handle these constraints. Two of the most commonly used methods are:

1. Treating the constraints explicitly (direct method)
2. Treating the constraints via penalty term (indirect method)

1.3.1 Treating constraint explicitly

This method is commonly used for gradient-based optimization and requires:

a) Objective function J and its gradient $dJ/d\mathbf{u}$
b) Constraint function $\mathbf{c}$ and its gradient $d\mathbf{c}/d\mathbf{u}$

If $\mathbf{u_k}$ is a feasible series of controls which satisfies all constraints at iteration k, then $\mathbf{u_{k+1}} = \mathbf{u_k} + \alpha_k \mathbf{g}$ is an improved feasible control for a sufficient small step $\alpha > 0$ if the vector $\mathbf{g}$ satisfies

$$\nabla J^T(\mathbf{u_k}) \cdot \mathbf{g} > 0 \text{ and } \nabla \mathbf{c}^T(\mathbf{u_k}) \cdot \mathbf{g} \geq 0 \qquad (3)$$

Because for $\alpha_k > 0$ small enough

$$J(\mathbf{u_{k+1}}) = J(\mathbf{u_k} + \alpha_k \mathbf{g}) \approx J(\mathbf{u_k}) + \alpha_k \nabla J^T(\mathbf{u_k}) \cdot \mathbf{g} > J(\mathbf{u_k}) \quad (4)$$

$$\mathbf{c}(\mathbf{u_{k+1}}) = \mathbf{c}(\mathbf{u_k} + \alpha_k \mathbf{g}) \approx \mathbf{c}(\mathbf{u_k}) + \alpha_k \nabla \mathbf{c}^T(\mathbf{u_k}) \cdot \mathbf{g} > \mathbf{c}(\mathbf{u_k}) \quad (5)$$

In output-constrained optimization, the constraint evaluation requires a simulation which needed to obtain the objective function value and its gradient. The difficulty here is to find the gradient of the constraints.

1.3.2 Treating constraints via penalty term

In this method, the constraint optimization is rephrased as an unconstrained optimization problem with only simple bounds on the control variables. The constraints are treated through adding a penalty term for each constraint to the objective function.

$\psi = J + \rho\, H(\mathbf{u})$ for sufficiently large penalty ρ gives a solution for the constraint optimization. It might be difficult to determine the right value for ρ. Approaches exist to update the penalty term ρ during the iteration process.

For more information on penalty term and treating constraints, see (Nocedal, 2006).

Chapter 2: Ensemble-based Optimization

2.1 Gradient-based Optimization

As already mentioned briefly in the introduction chapter, there are several methods to do optimization process, e.g. gradient-free methods and gradient-based methods. The gradient-free methods have one major weakness; it is very computational expensive for a large number of variables. One way to deal with large number of variables is by using the gradient-based method instead. Since gradient information is required to do gradient-based optimization and analytic gradients in general are not always available, approximate gradients is an alternative way to provide the gradient information. Approximate gradient information of the objective function with respect to the control variables can be provided by several ways. Two of the most common ways to obtain an approximate gradient is the finite difference method and ensemble-based method.

The basic idea of using gradient information is to use the gradient to iteratively update the control variables with improved objective function values. There are two common strategies on how to update the control variables: the line search strategy and the trust region strategy. Unfortunately, gradient-based optimization has a tendency to get stuck in a local optimum point and have difficulty in finding the global optimum point.

In addition to the gradient-based optimization methods, there are other methods that are less likely to be trapped in a local maximum (or minimum) and that can be used with nondifferentiable objective functions. However, they typically converge slowly and become inefficient when the number of variables to be optimized is large (Dehdari, 2011). Harding et al. (1988) evaluated several approaches to optimize production scheduling including the genetic algorithm (GA), simulated annealing, Sequential Quadratic Programming (SQP), and several hybrid approaches. In their study, the GA method achieved significantly better result than the other methods for random starting points. They noted that the SQP method performed poorly on a problem that required adjusting the starting and ending times as the objective function was discontinuous with respect to those variables.

2.1.1 Line Search Strategy

The idea of line search strategy is to determine first the direction of the control update and secondly, to determine how far to step along that direction. This can be done by searching along this direction $\mathbf{d_k}$ from the current iterate $\mathbf{u_k}$ for a new iterate with higher function value. If the step-length taken in that direction results in a non-increasing objective function value, it means that the length of the step is not correct, usually because the chosen step length is too large. The step length then needs to be adjusted, usually by making it smaller. This process is also known as backtracking.

For more information on line search strategy, see (Nocedal, 2006).

2.1.2 Trust Region Strategy

The idea of trust region strategy is to determine the step length or the trust region size first and then to determine the direction of the control update. This can be done by constructing a model function m_k whose behavior near the current point $\mathbf{u_k}$ is similar to that of the actual objective function J. After that, the search for an optimum of the model function m_k is restricted to some region around $\mathbf{u_k}$:

$$\max_{p} \; m_k(\mathbf{u_k} + \mathbf{d}); \quad \mathbf{u_k} + \mathbf{d} \text{ lies inside the trust region} \tag{6}$$

If the result is not satisfying, then it means the trust region radius is too large and one should make it smaller and recalculate the optimization. If the model is acceptable, the radius for the next step will be the same or even expanded.

but most of the time the model m_k is approximated only until the first order for simplicity reason and also to reduce the computational-cost. It is important to note that J_k is a scalar and ∇J_k is a vector.

$$m_k(\mathbf{u_k} + \mathbf{d}) = J_k + \mathbf{d}^T \nabla J_k \tag{7}$$

For more information on trust region strategy, see (Nocedal, 2006).

2.1.3 Outer and Inner Iteration

In gradient-based optimization, the optimum solution is obtained through an iteration process. There are two main iteration processes involved in the optimization process: an outer iteration and an inner iteration.

Outer iteration is the process of updating the current control using the gradient of the objective with respect to the control parameters. The gradient is used to determine the direction of the update. The inner iteration is the process aiming at re-determining a proper trust region radius or step length that results in an update step with an (acceptable) increase in the objective function. Both the outer and inner iteration will continue to run until it finds the maximum points of the objective function, reaches the maximum number of iteration allowed, or violates a certain prescribed tolerance, e.g. a tolerance on the control change or on the objective function change.

For more information on outer and inner iteration, see (Nocedal, 2006).

2.2 Ensemble-based Gradient Calculation

2.2.1 Description and Gradient Calculation

The ensemble-based optimization (EnOpt) method is a relatively new method and was proposed by Lorentzen et al. (2006) and Nwaozo (2006), and further developed by Chen (2008) and Chen et al. (2009) who presented the method as mostly used today. Several other, more recent publications about the method are Masoor et al. (2009), Chen and Oliver (2010), Su and Oliver (2010), Leeuwenburgh et al. (2010), Chen and Oliver (2012), and Fonseca (2013).

Ensemble-based optimization is an optimization method to obtain a stochastic gradient of the objective function with respect to the control variables. This gradient is approximated by evaluating the objective function values for an ensemble of control vectors chosen from a multi Gaussian random distribution with known mean and covariance matrix. By using the ensemble-based gradient, the number of simulation required to obtain a stochastic gradient is nearly independent to the number of control variables although it is expected that for an increasing number of control variables more perturbations are needed to obtain a sufficiently accurate gradient. Furthermore, the ensemble-based optimization is independent of the reservoir simulator and its solver, as the simulator can be treated as a blackbox (Chen, 2008). This gives great flexibility to use different reservoir simulators.

The method has been shown to achieve good results for a variety of different reservoir models even though it has lower computational efficiency and accuracy compared to adjoint method. The efficiency of the method, however, depends on the size of the ensemble and the level of nonlinearity of the problem.

The control variables can be stored as a vector denoted as $\mathbf{u}$ and is defined as follows

$$\mathbf{u} = [u_1 \;\; u_2 \;\; \dots \;\; u_N]^T \tag{8}$$

where N is the total number of control variables and proportional to the number of controls, number of wells, and number of control time steps. To estimate the gradients, a mean-shifted ensemble matrix and mean-shifted objective function vector are defined as

$$\tilde{\mathbf{U}} = [\mathbf{u_1} - \bar{\mathbf{u}} \;\; \mathbf{u_2} - \bar{\mathbf{u}} \;\; \cdots \;\; \mathbf{u_M} - \bar{\mathbf{u}}]^T \tag{9}$$

and

$$\tilde{\mathbf{j}} = [J_1 - \bar{J} \;\; J_2 - \bar{J} \;\; \cdots \;\; J_M - \bar{J}]^T \tag{10}$$

where

$$\bar{\mathbf{u}} = \frac{1}{M}\sum_{m=1}^{M} \mathbf{u}_m \quad and \quad \bar{J} = \frac{1}{M}\sum_{m=1}^{M} J_m \tag{11}$$

are the ensemble's control variables and objective function mean, respectively, with M the ensemble size and m the ensemble index number.

For an overdetermined case, where $M > N$, the approximate gradient g with respect to the controls could be obtained as a least squares solution:

$$\mathbf{g} = \left(\tilde{\mathbf{U}}^{\mathbf{T}}\tilde{\mathbf{U}}\right)^{-1}\tilde{\mathbf{U}}^{\mathbf{T}}\,\tilde{\mathbf{j}} \tag{12}$$

where in practice it is computationally more efficient to solve a linear system of equations for gradient $\mathbf{g}$ rather than computing the inverse. The equation above for gradient g can also be expressed as

$$\mathbf{g} = \mathbf{C}_{\mathbf{uu}}^{-1}\,\mathbf{C}_{\mathbf{uj}} \tag{13}$$

where

$$\mathbf{C}_{\mathbf{uu}} = \frac{1}{M-1}\left(\tilde{\mathbf{U}}^{\mathbf{T}}\tilde{\mathbf{U}}\right) \;\; and \;\; \mathbf{C}_{\mathbf{uj}} = \frac{1}{M-1}\left(\tilde{\mathbf{U}}^{\mathbf{T}}\tilde{\boldsymbol{j}}\right) \tag{14}$$

are the ensemble (sample) covariance and cross-covariance matrices respectively (Chen, 2008, and Chen and Oliver, 2009). The derivation for the linear regression formula (12) can be found in many introductory linear algebra textbooks, e.g. Strang (2006).

For more common cases where the problem is underdetermined, that is $M < N$, the matrix product $\bar{\mathbf{U}}^{\mathbf{T}}\bar{\mathbf{U}}$ is rank deficient and one cannot directly compute the inverse or solve the associated system of equations because the matrix has to be full rank and non-singular (Fonseca, 2013). This limitation can be solved by decomposing the large matrix using singular value decomposition (SVD); see e.g. Strang (2006). Alternatively, Chen (2008) and Chen and Oliver (2009) proposed to simply use

$$\mathbf{g}' = \mathbf{C}_{\mathbf{uj}} = \mathbf{C}_{\mathbf{uu}}\,\mathbf{g} \tag{15}$$

instead of the gradient $\mathbf{g}$ by the ensemble cross covariance $\mathbf{C}_{\mathbf{uj}}$, or even

$$\mathbf{g}'' = \mathbf{C}_{\mathbf{uu}}\,\mathbf{C}_{\mathbf{uj}} = \mathbf{C}_{\mathbf{uu}}\,\mathbf{C}_{\mathbf{uu}}\,\mathbf{g} \tag{16}$$

where the second premultiplication with $\mathbf{C}_{\mathbf{uu}}$ works as a preconditioning step.

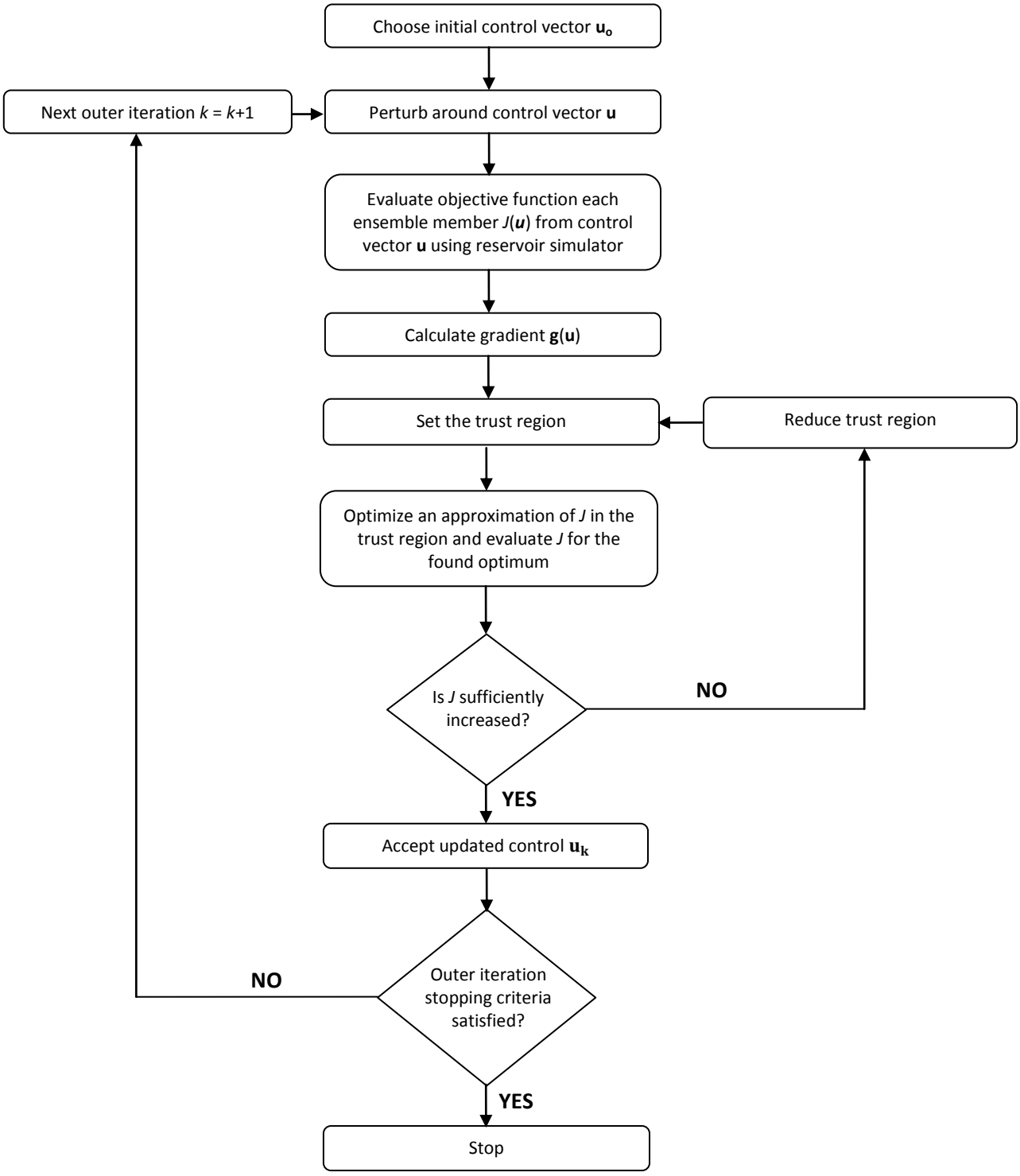


Figure 5. Flowchart for ensemble optimization procedure

2.2.2 Gradient Regularization

Regularization is a feature in optimization to correlate neighboring time step controls with the current time step control so that they are affecting each other. By using the gradient regularization, the optimized controls will have a smoother profile. There are 3 options available in the regularization option, named Regularization 2 until Regularization 4. Regularization option 1 means no correlation or smoothing process is being applied and it is set as the default setting. Regularization option 2 and 3 are using Gaspari function to obtain the premultiplicator matrix for the objective function gradient where the Gaspari function is expressed as follows:

$$Gs(z,ts) = \begin{cases} -\frac{1}{4}\left(\frac{z}{c}\right)^5 + \frac{1}{2}\left(\frac{z}{c}\right)^4 + \frac{5}{8}\left(\frac{z}{c}\right)^3 - \frac{5}{3}\left(\frac{z}{c}\right)^2 + 1 \;;\; \text{for} \;\; z < c \\ \frac{1}{12}\left(\frac{z}{c}\right)^5 - \frac{1}{2}\left(\frac{z}{c}\right)^4 + \frac{5}{8}\left(\frac{z}{c}\right)^3 + \frac{5}{3}\left(\frac{z}{c}\right)^2 - 5\left(\frac{z}{c}\right) + 4 - \frac{2}{3}\left(\frac{z}{c}\right)^{-1} \;;\; \text{for} \;\; z < 2\,c \\ 0 \;\; ; \;\; \text{for}\; z > 3\,c \end{cases} \tag{17}$$

$$\mathbf{C_{uu}}(i,j) = Gs \tag{18}$$

$$z = abs(i-j) \tag{19}$$

where

Gs = Gaspari correlation function

$\mathbf{C_{uu}}$ = premultiplication matrix

z = distance between control time steps

c = time correlation

i, j = control time step indices

For regularization option 2, the gradients are multiplied by the Gaspari correlation matrix Gs once and for regularization option 3, the (original) gradients are multiplied by the Gaspari correlation matrix Gs twice as preconditioning step. This regularization option 2 and 3 are the same as the preconditioning step proposed by Chen (2008) and Chen and Oliver (2009) in equation (15) and (16).

Regularization option 4 however, is slightly different than the previous two regularizations. This regularization option tries to smooth the gradient using Gaspari correlation function with some modification as follow:

$$nc = 4\,c + 1 \tag{20}$$

$$\mathbf{k} = [1\ 2\ 3\ \ldots nc] \tag{21}$$

$$\mathbf{c}(\mathbf{k}) = Gs\big(abs\big(\mathbf{k} - (2\,c+1)\big), c\big) \tag{22}$$

$$\mathbf{c1}(i) = c(\max(1\,, 2\,c + 2 - i) : \min(nc\,, 2\,c + 1 + \; n - i)) \tag{23}$$

$$\mathbf{sc1}(i) = sum(\mathbf{c1}(i)) \tag{24}$$

$$\mathbf{P}(i,j) = \mathbf{c1}(i) \times \frac{\mathbf{g}(\max(1, i - 2\,c) : \min((n), i + 2 + c), j)}{\mathbf{sc1}(i)} \tag{25}$$

Where

Gs = Gaspari correlation function

c = time correlation

n = control time steps, n = 10

i = control time steps indeces, for i = 1 to 10

j = well indeces, for j = 1 to 4

An example of the difference between regularized and unregularized strategies for output-constrained optimization with ICV as the controls can be seen in figure 6 below:

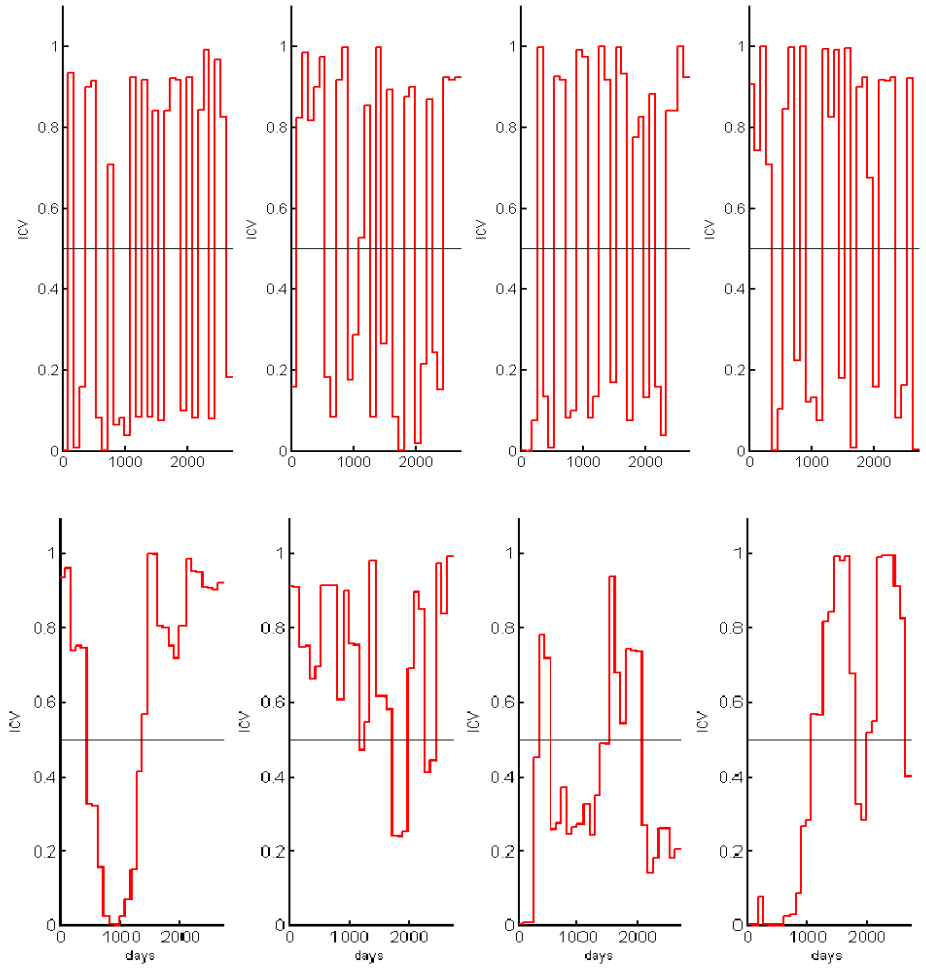


Figure 6. Comparison between unregularized (above) vs regularized (below) for 30 controls

Chapter 3: Bound- and Input-Constrained Optimization

3.1 Model Description

A small reservoir model adapted from (Kraaijevanger et al. 2007) was created with a sealing L-shaped fault, 20% porosity, 100 mD permeability in X and Y direction, and 10 mD permeability in Z direction. The reservoir model is a single-layer model, with 4 injectors in the corners of the reservoir and 1 producer in the middle of the reservoir. The economic parameters are: discount factor 0%, oil price \$125/m^3, water production cost \$25/m^3, and water injection cost \$5/m^3. This reservoir model is depicted in Figure 2.

Both the trust region optimizer and Matlab *fmincon* optimizer are set to evaluate the model using 10 control time steps for 2700 days with injection rates as the control. For the constrained case, the field injection rate is selected as the constraint and is limited to 600 m^3/day while each injector has a maximum injection capacity of 250 m^3/day and a minimum injection rate of 0 m^3/day. The initial injection rate used is 100 m^3/day and this is a feasible reference strategy. In other words, the optimization is started from feasible region.

The gradient is approximated using ensemble with 15 samples and simulation time step size of 90 days to obtain the estimate gradient. The perturbation size of 10 m^3/day is used with no correlation is imposed between the controls over time. There is also no preconditioning nor smoothening process applied to the gradient. The reservoir simulator used in this simulation is an open source simulator developed by Sintef (Norway) called MRST or Matlab Reservoir Simulation Toolbox (Lie et al. 2010, 2012).

For trust region optimizer, the maximum number of inner iterations allowed is 3. The trust region radius is initialized at 0.2 and has a maximum trust region radius of 0.2. The trust region radius is not reinitialized after each outer-iteration, which means that during new outer iteration, the initial trust region radius starts from the previous radius value. The trust region radius tolerance is set at 0 and has trust ratio 1 of 0.0001 and trust ratio 2 of 0.1. These trust ratios indicate the criteria for the ratio between the actual and predicted increase in the objective function value. The trust region has contraction factor of 0.5 and expansion factor of 2. All simulation runs, both with *fmincon* and trust region, have a maximum number of outer iterations of 50 as the stopping criterion.

The control variables u_k^j are the injection rates for injectors $j = 1, \dots, 4$ for each control time step $k = 1, \dots, 10$. The objective function J is the NPV where:

$$\max J(\mathbf{u}) = \sum_{k=1}^{N} \left[\frac{\{q_{o,k}\, r_o - q_{wp,k}\, r_{wp} - q_{wi,k}\, r_{wi}\}\, \Delta t_k}{(1+d)^{\frac{t_k}{365}}} \right] \tag{26}$$

Subject to:

Table 1. Bound and Input Constraints

$0 \leq \mathbf{u}_k^j \leq 250$	Bound constraints
$\mathbf{u}_k^1 + \mathbf{u}_k^2 + \mathbf{u}_k^3 + \mathbf{u}_k^4 \leq 600$ for $k = 1$ to 10	Input Constraints

Where:

$J(\mathbf{u})$ = NPV as the objective function

$q_{o,k}$ = oil production rate for simulation time step k

$q_{wp,k}$ = water production rate for simulation time step k

$q_{wi,k}$ = water injection rate for simulation time step k

r_o = oil price

r_{wp} = water production cost

r_{wi} = water injection cost

Δt_k = simulation time step length

t_k = simulation time

d = discount factor

k = simulation time step index

N = total number of simulation time steps

3.2 Reference Case: Optimization with Heuristic Rule

Both the simulator and the optimizer can handle constraints, but in a different way. Commercial simulators provide a simple heuristic algorithm that does not take the maximization of the objective function into consideration, while the optimizer will handle the constraints by taking the maximization into consideration. This difference will make the constraint-handling done by the simulator suboptimal compared to the constraint-handling done by the optimizer.

The heuristic rule that has been implemented in the experiments is based on proportional reduction. First, it will calculate the constraint violation or the excess of the liquid injection as can be seen on equation (27). This excess calculation will be used to find the proportional reduction needed to decrease the initial controls (e.g. the injection rates) so that the new controls will satisfy the constraints as shown on equation (28) below.

$$\varepsilon = \sum_{i=1}^{n} \mathbf{u_i} - c_{max} \tag{27}$$

$$\mathbf{u_{i,new}} = \mathbf{u_i} - \frac{\varepsilon}{\sum_{i=1}^{n} \mathbf{u_i}} \mathbf{u_i} \tag{28}$$

Where:

ε = excess

$\mathbf{u_i}$ = controls (e.g. injection rates)

$\mathbf{u_{new}}$ = updated controls based on the heuristic rule

c_{max} = input constraint (e.g. maximum field injection rate)

i = well number

n = total number of injection wells

These calculations are done at each simulation time steps, which means the control strategy becomes linked to the simulator time stepping. This heuristic rule will allow the simulator to satisfy the constraints, but the objective function will not be optimal because the simulator will not consider other strategies than the prescribed heuristic algorithm. The optimizer, however, will try to use the gradient of the constraints and the objective function with respect to the control variables and see what the best way to address these constraints is. This way, the optimizer will find a solution that is more optimal than what the simulator does.

3.3 Results and Discussion

3.3.1 Bound Constraints with Ensemble Gradients

Bound constrained optimization is not unconstrained optimization, but algorithms for unconstrained optimization can easily be adapted to deal with bound constraints. As already described in model description subsection above, the boundaries for the injection rate controls are the minimum and maximum injection rates. In this case, the minimum injection rate is 0 m^3/day which means the injection well cannot be transformed into a production well, and the maximum injection rate allowed is 250 m^3/day.

Figure 7, 8, 9, and 10 below show different injection rates strategies with different optimization methods as follow: SQP, Interior Point, Active Set, and trust region. From the four figures below, it can be seen that the four optimization methods have no problem in satisfying the bound constraints.

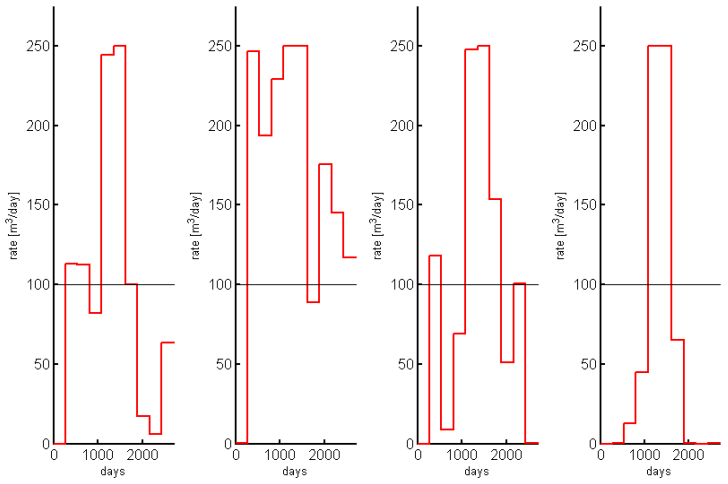


Figure 7. Injection rates for bound-constrained optimization with SQP method

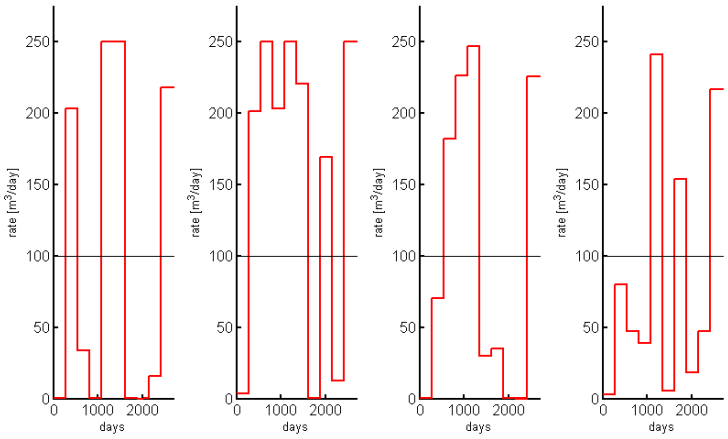


Figure 8. Injection rates for bound-constrained optimization with Interior Point method

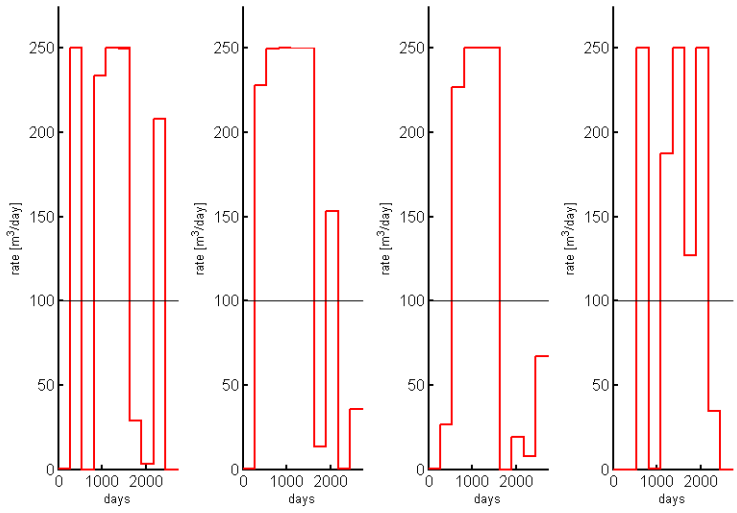


Figure 9. Injection rates for bound-constrained optimization with Active Set method

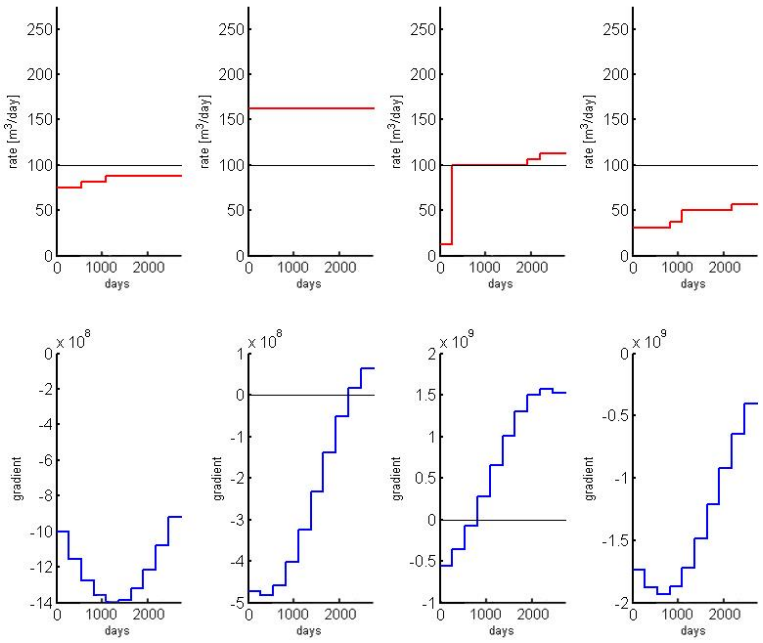


Figure 10. Injection rates for bound-constrained optimization with trust region strategy

The lower panel of figure 10 shows the gradient of the objective function with respect to its control variables. The (local) optimum solution is obtained when the gradient shows zero values or close to zero values. This gradient information will provide direction on how the next control perturbation should be done. Positive gradient means the direction on how control changes is the same with the direction on how the objective function changes, and vice versa for negative gradient value.

Different methods give different objective functions values as shown in figure 11. It can be seen that the trust region optimizer performs better than the other three optimizers for bound-constrained optimization problem in terms of the final objective function value obtained and also the computational efficiency. Both figures 11 and 12 show that in terms of outer iteration numbers and also function evaluation numbers, the trust region requires the least effort but gives the highest NPV. The SQP method performs second best, both in terms of the objective function value and the computational efficiency.

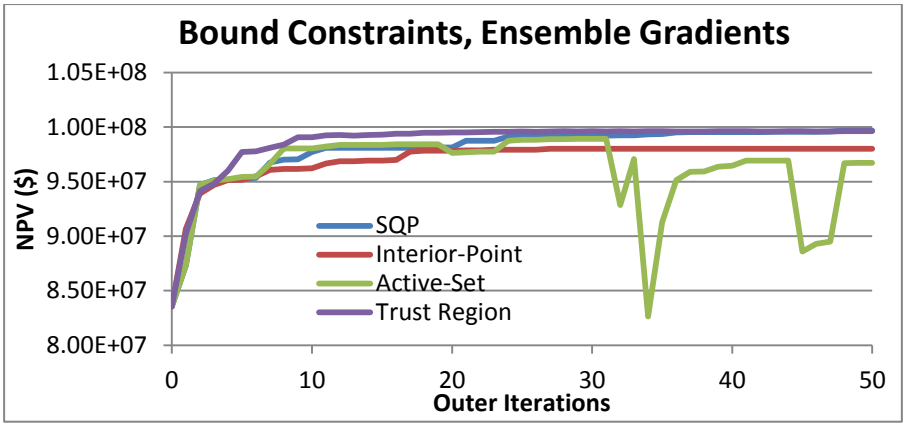


Figure 11. Bound constraints with ensemble gradient for outer iterations vs NPV

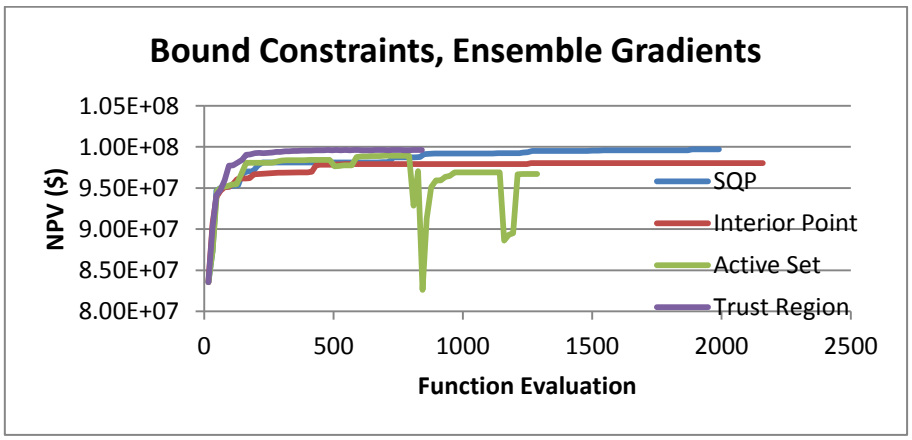


Figure 12. Bound constraints with ensemble gradient for function evaluation vs NPV

From figures 11 and 12, it can be seen that the trust region strategy performs better than the other three more advanced optimizers. This result seems counterintuitive because the three more advanced optimizers are expected to perform better than the simpler trust region strategy. In the beginning it was thought that the difference in Hessian matrix implementation caused this. It is expected that the Hessian matrix could give a better accuracy for the optimization process, but since the Hessian matrix is just an approximation of the second derivative, it is possible that the Hessian matrix would give error and make the optimization process perform worse than the one without the Hessian matrix.

To test this hypothesis, bound-constrained optimization with the Interior Point method without implementing the Hessian matrix was run. The same test with SQP or Active Set could not be run because of *fmincon* architecture. *Fmincon* prevents the user to run the optimization without the Hessian matrix for SQP and Active Set methods. Figure 13 below shows the comparison between the trust region, Interior Point with Hessian, and Interior Point without Hessian. As can be seen, even though the Interior Point without Hessian gives a higher objective function value than the Interior Point with Hessian, but both results still perform worse than the trust region strategy. In other

words, the Hessian matrix implementation probably is not the reason why the three more advanced optimizers do not perform as good as the trust region.

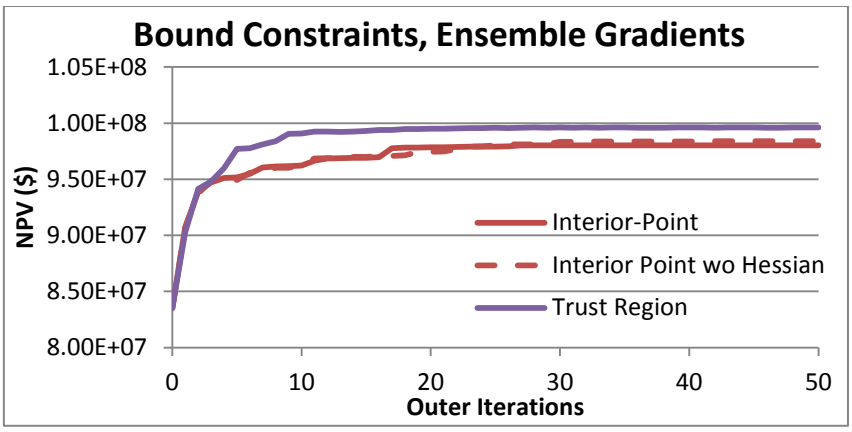


Figure 13. NPV comparison for bound-constrained optimization to see the effect of Hessian matrix

Although it is not entirely clear what caused this, but it is possible that the trust region strategy is more robust to errors in the gradients because it focuses only on the sign of the gradient elements and set very small values to 0.

3.3.2 Input Constraints with Ensemble Gradients

In this input-constrained optimization problem, a limitation of total injection rate of 600 m^3/day is placed to constrain the injection rates as the well controls. All injector wells still have the minimum and maximum injection rate bounds of 0 and 250 m^3/day. With this input constraints, now the optimizers have to satisfy both the bounds and the input constraints.

In figures 14, 15, 16, 17, and 18, only three optimizers are compared because the trust region optimizer cannot be used for input-constrained optimization problems. The TNO trust region optimizer has not been designed to handle input- or output-constrained optimization problems.

In figures 14, 15, and 16, the production rate profile from the reservoir can be observed. The black-colored line indicates the liquid production rate which equals to the total of water and oil production rates. The water production rate is indicated by the blue-colored line and oil production rate is indicated by the red-colored line. As can be seen from the three figures, all three optimizers can fulfill the given input constraints.

From both figures 17 and 18 it can be seen that, during early iterations, the Interior Point appears to perform better than the other two optimizers in terms of NPV, but that after the 41st outer iterations, the objective function starts decreasing until the 46th outer iteration and then it starts increasing again until the optimization process stops after fifty outer iterations because the optimization process reaches the stopping criterion. On the other hand, the SQP method, which

performed better than the Interior Point method during the bound constraint optimization problem, gives lower objective function values during early iterations but shows no decrement during the optimization process. Active Set always performs the worst of all optimizers in terms of the computational efficiency, the objective function, and also the stability. Both in bound- and input-constrained cases, the Active Set fluctuates which is not good because it means the Active Set is not a stable optimizer for this problem.

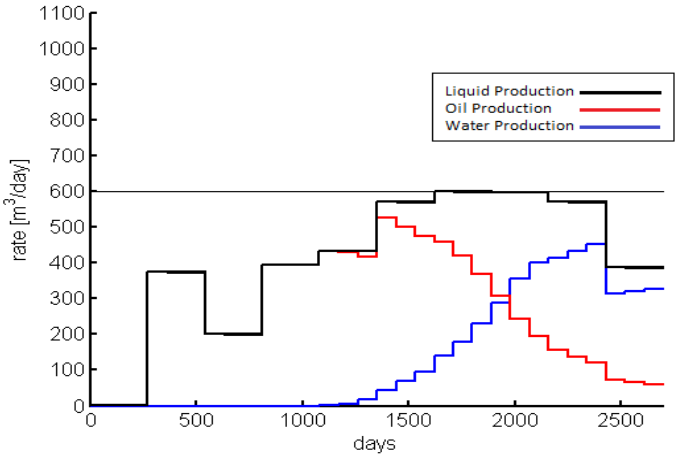


Figure 14. Production rate profile of input-constrained optimization with SQP

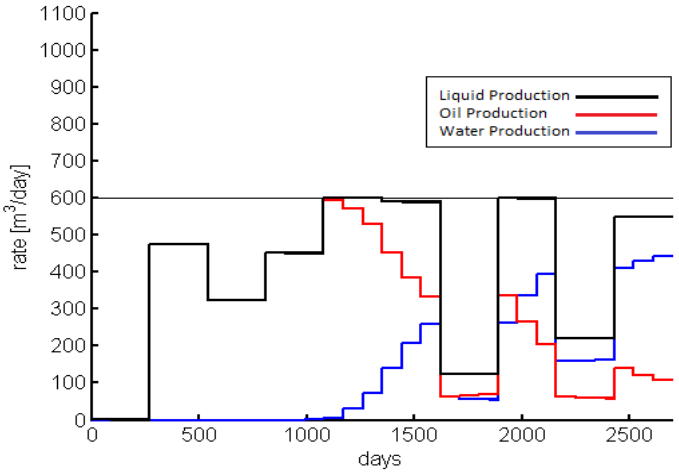


Figure 15. Production rate profile of input-constrained optimization with Interior Point

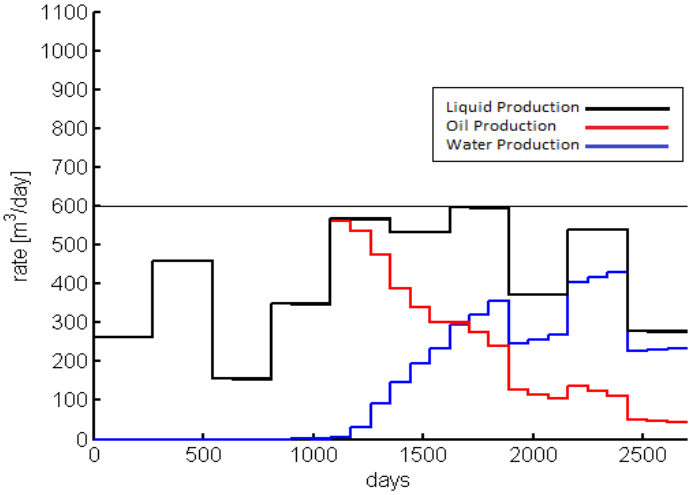


Figure 16. Production rate profile of input-constrained optimization with Active Set

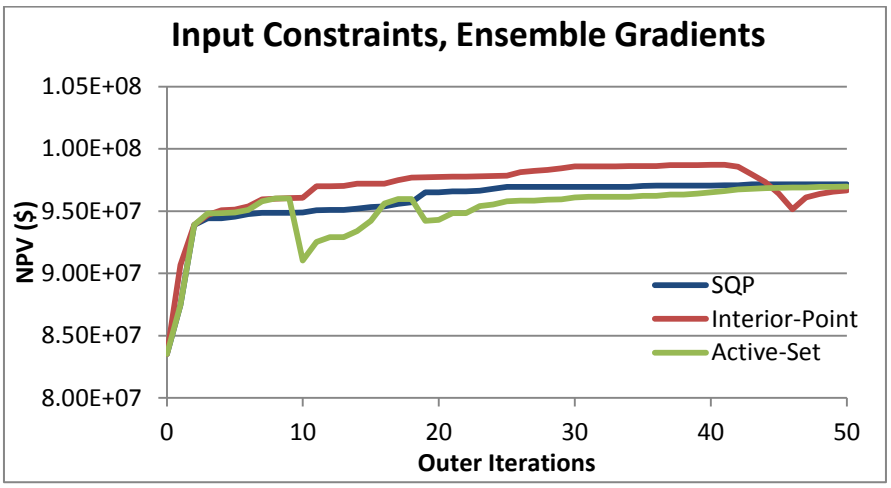


Figure 17. Input constraints with ensemble gradient for outer iterations vs NPV

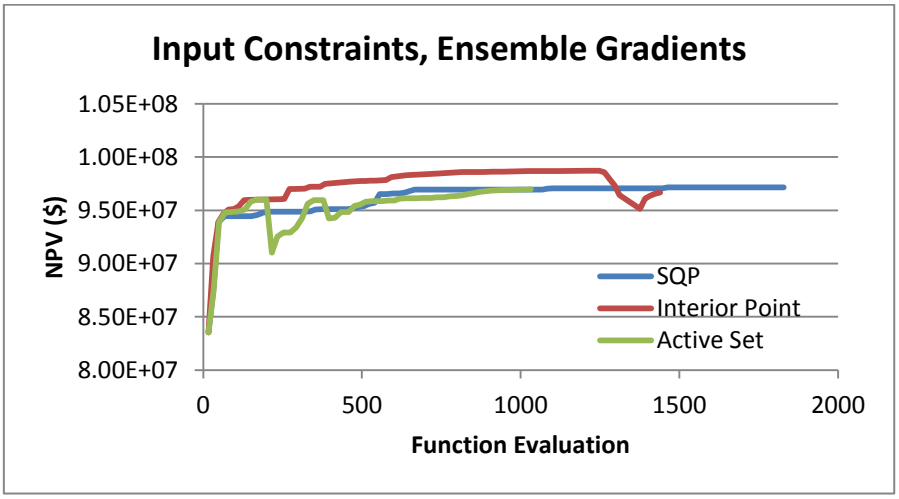


Figure 18. Input constraints with ensemble gradient for function evaluation vs NPV

The following discussion will focus on what caused this objective function decrement. The inconsistent occurrence of this decrement also made it more difficult to look for similarities between these cases to find a pattern where or when this decrement will happen, e.g. it happened on Active Set optimizer during the bound- and input-constrained optimization with ensemble gradients (figures 11, 12, 17, and 18). This decrement on Active Set can also be observed later on in the next subchapter in the bound-constrained optimization with the finite difference gradients (figures 20 and 21) but not observed with the input-constrained optimization (figures 22 and 23). This decrement also happened on Interior Point but only during the input-constrained optimization while none of this decrement happened on SQP.

After studying the diagnostic of the results, an interesting observation was made. Whenever the objective function decreases, the number of constraint violations also decreases. This observation can be seen in figure 19 below. The continuous blue and red lines are showing NPV while the dashed blue and red lines are indicating the constraint violation, using the secondary y-axis on the right side. As it can be seen, there are constraint violations with the Interior Point first 45 outer iterations. This means that even though the Interior Point method seems to be performing better than SQP, actually it was the other way around. Interior Point's higher NPV was obtained by the infeasible solution while the SQP can maintain its feasibility since the first iteration. The Interior Point was still looking for the feasible region while improving the objective function at the same time. On the other hand, the SQP solution was already inside the feasible region and thus only needed to improve the objective function. By considering the feasibility during the iteration process, the final objective function, and also the number of iterations required, it can be said that in this particular case, the SQP performs better than the Interior Point.

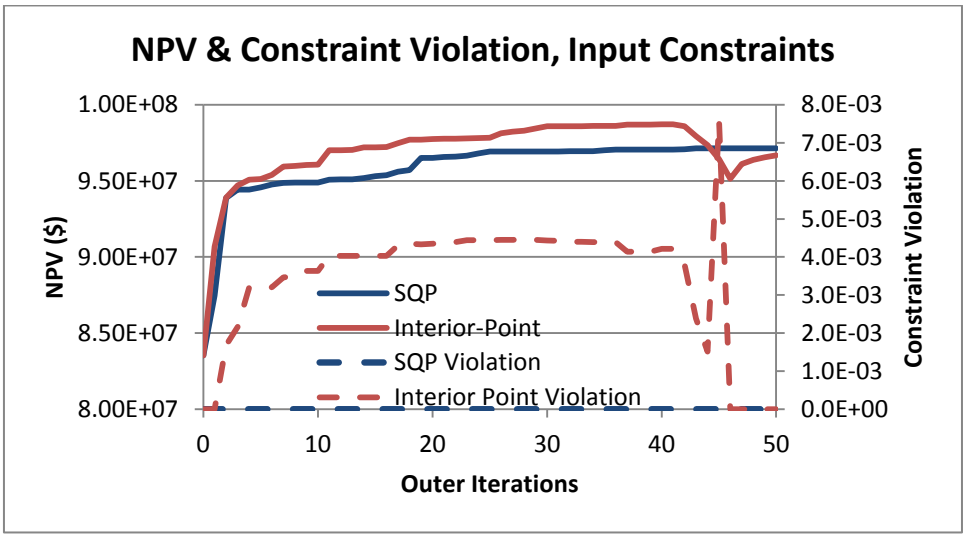


Figure 19. NPV and constraint violation vs outer iterations for SQP and Interior Point optimizers

3.3.3 Bound Constraints with Finite Difference Gradients

As already explained in Chapter 2.2, the ensemble gradient method is a relatively new method that was proposed as a substitute for the finite difference method. This method offers the ability to provide the approximate gradient with similar quality to the finite difference method but with less function evaluations because it is nearly independent of the number of control variables.

In this subchapter, both the finite difference gradient method and the ensemble-based gradient method will be compared using the bound-constrained optimization problem with the same reservoir model. The results can be seen in Figure 20 and 21 below.

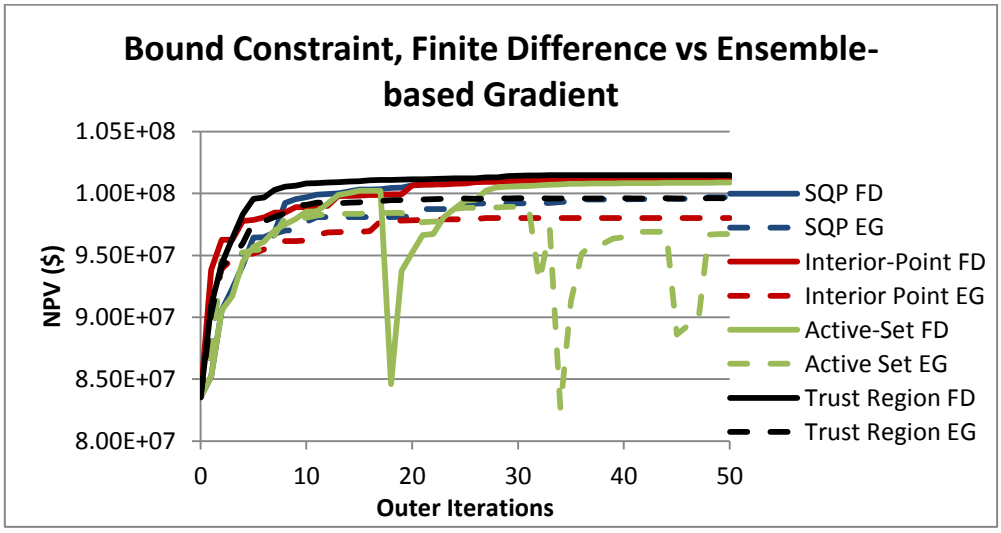


Figure 20. NPV vs outer iterations for finite difference and ensemble-based gradients in bound-constrained optimization

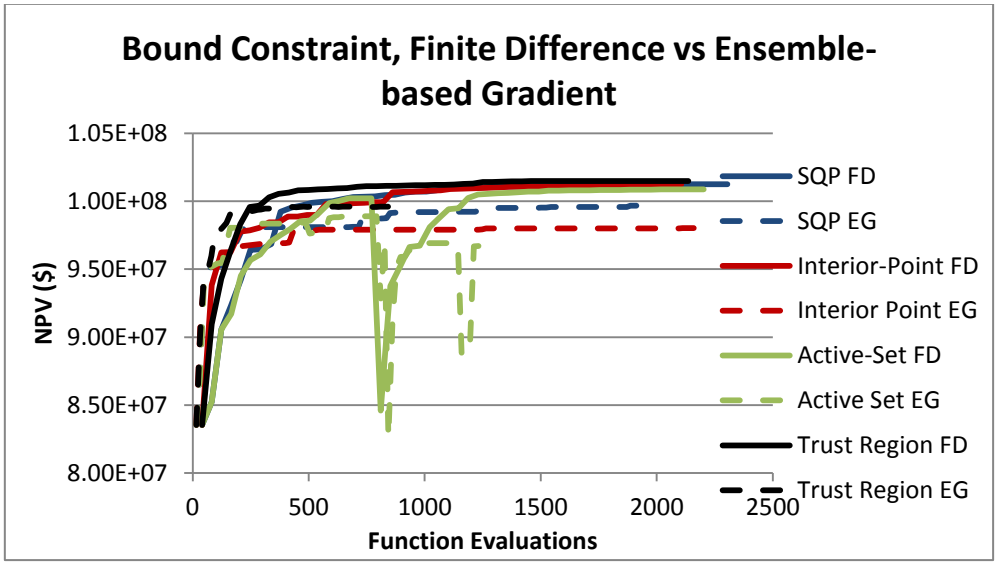


Figure 21. NPV vs function evaluations for finite difference and ensemble-based gradients in bound-constrained optimization

Similar to the ensemble-based gradient results, the trust region strategy with finite difference gradient (figure 20 and 21) performs better than the other three methods in terms of the computational efficiency and objective function while the other three optimizers perform relatively similar to another.

From figure 20, it can be seen that all results using a finite difference method, indicated by continuous lines, perform better than the ensemble gradient, indicated by dashed lines. This result is expected because the finite difference gradient is more accurate than the ensemble gradient and thus closer to the true gradient, at the cost of being more computationally expensive. The finite difference gradient requires $N + 1$ simulations in order to obtain the gradient with N is the number of control variables while the ensemble gradient only needs M simulations with M is the size of ensemble and it is user-defined. If the chosen ensemble size M is less than the control variables N, then the ensemble gradient requires less simulations to obtain the gradient. In this case, there are 40 control variables (*N*) and 15 samples (*M*).

3.3.4 Input Constraints with Finite Difference Gradients

Just like in the previous chapter, a comparison between the finite difference gradient and the ensemble-based gradient for input-constrained optimization problem was also done and the results can be seen in figures 22 and 23 below.

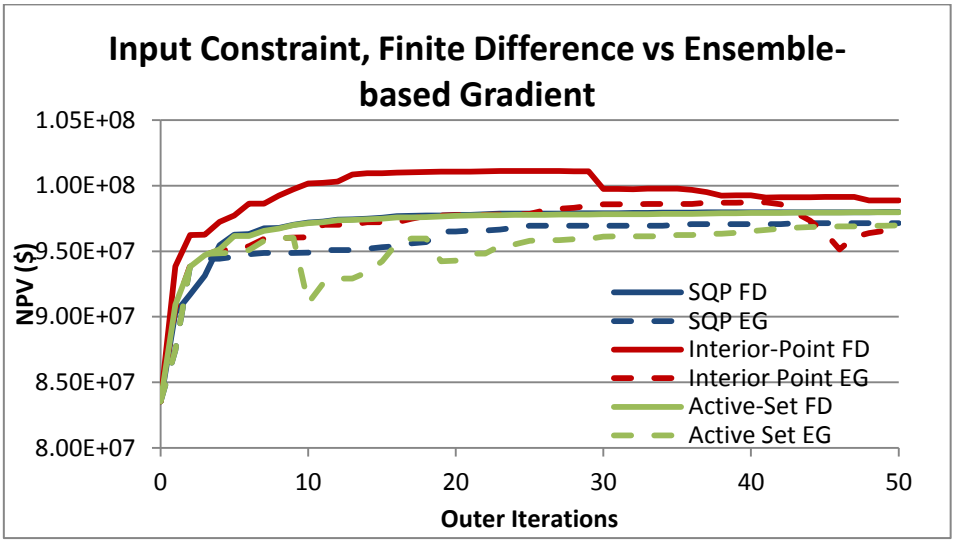


Figure 22. NPV vs outer iterations for finite difference and ensemble-based gradients in input-constrained optimization

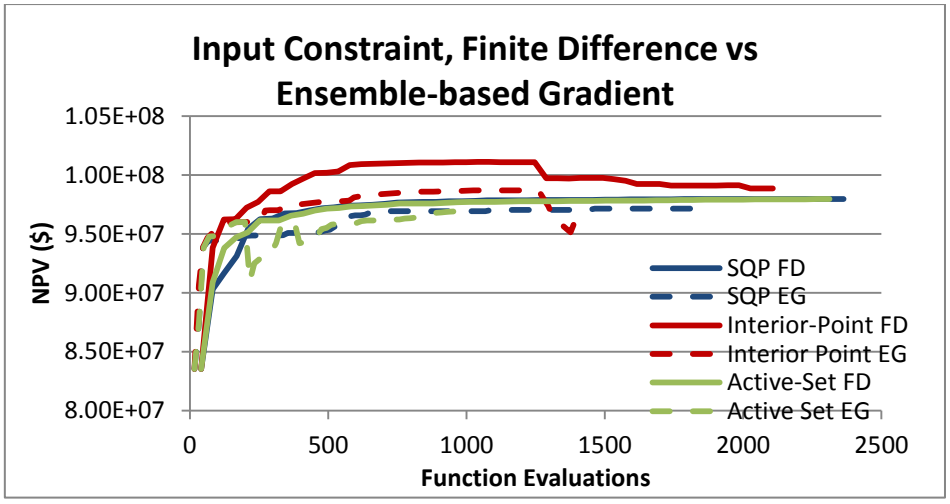


Figure 23. NPV vs function evaluations for finite difference and ensemble-based gradients in input-constrained optimization

As for the input-constrained optimization problem, the results are also quite similar to the ensemble gradient results: the Interior Point seems to perform better than the other two optimizers during the early iterations while the SQP and Active Set have more or less similar performance. However, in this problem, Interior Point again is showing the same issue as occurred in the bound-constrained problem: a decrement of the objective function during the optimization process. As explained before, this is because the Interior Point was still in the infeasible region and thus it is just logical to have a higher NPV than the SQP which is in the feasible region. Along the process, the Interior Point was trying to reduce the constraint violations to move from the infeasible region to feasible region. This process leads to the decrement in the objective function instead of an increment. Once the Interior Point has moved inside the feasible region, it will start increasing the objective function. It can be concluded that SQP and Active Set perform equally well and better than Interior Point for this particular case.

3.3.5 Input Constraints with Ensemble Gradients Using 50 Samples

From the plots and discussion above, it is observed that the finite difference gradients perform better than the ensemble gradients, but this is understandable because of the difference in function evaluations. By using the finite difference gradients, there will be 41 function evaluations per outer iteration (because there are 40 control variables) while there will be only 16 function evaluations per outer iteration for the ensemble gradients (because are only 15 samples). The idea of testing ensemble gradients with 50 samples is to see whether using more function evaluations (by having more samples) than the finite difference gradients can provide better results in terms of the objective function.

This ensemble gradient with 50 samples test is conducted for input-constrained optimization, comparing the three optimizers: SQP, Interior Point, and Active Set. The result of the test can be seen in figure 24 below.

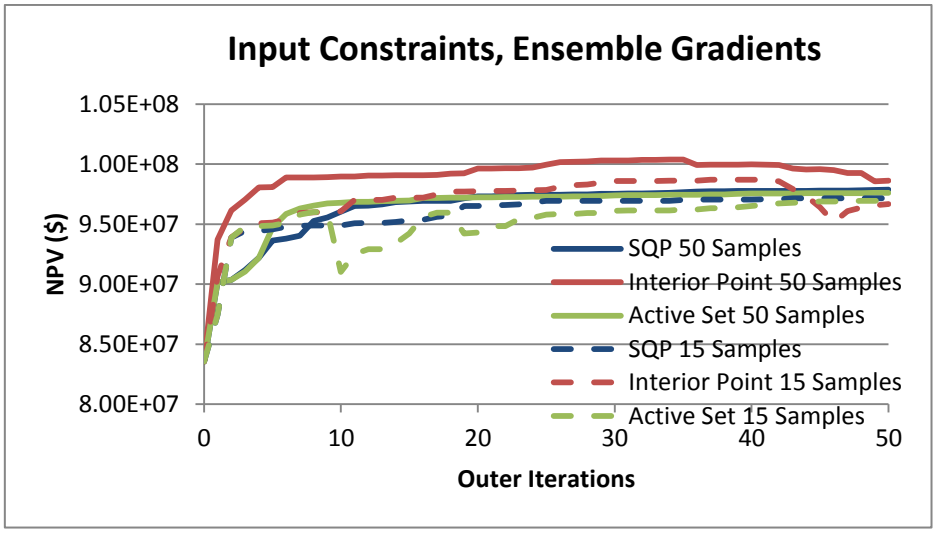


Figure 24. NPV vs outer iteration for ensemble gradients with 15 and 50 samples

From the figure above, it can be observed that indeed the results using 50 samples give higher objective functions than the ones using 15 samples. This is understandable because as what already discussed above, using more samples means more function evaluations and will result in more detailed gradient, better optimization and higher objective functions. Next, these results will be compared with the finite difference gradients results. The comparison is shown in figure 25 below. Initially, it could have been expected that the ensemble gradients with 50 samples would give a higher objective function value because it has more function evaluations, but apparently that was not the case. The finite difference gradients still give higher objective function values than the ensemble gradients. These results are most likely caused by the difference in how the gradients are obtained. The finite difference method is more systematic in obtaining the approximate gradients by perturbing the sample control one-by-one, but less efficient than the ensemble gradients that perturb all the sample controls at the same. Therefore, even though the finite difference has less function evaluations, it still has better-quality gradients approximation.

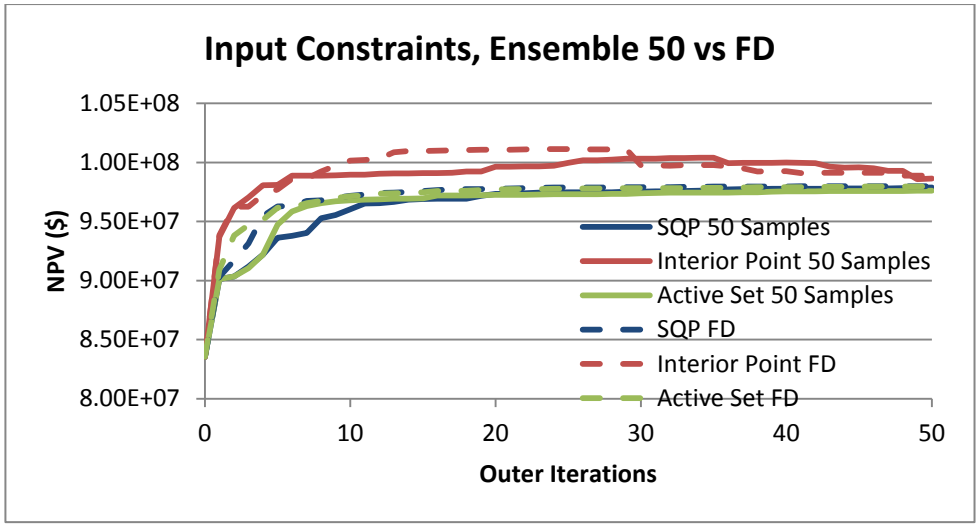


Figure 25. NPV vs outer iteration for ensemble gradient with 50 samples and finite difference gradient

3.3.6 Input Constraints Handling Between Optimizer and Simulator

As already described in chapter 3.2, commercial simulators can also handle input constraints by implementing some heuristic rules explained. Figure 26 below shows the comparison between input constraint handling by the optimizers and the simulators. The solid lines in figure 26 for SQP and Interior Point are the same as the solid lines in figure 19. It means that the Interior Point solution is infeasible for the first 45 outer and the decrement observed is caused by the optimizer trying to improve the solution from infeasible into feasible region and thus making the comparison between the optimizer and the simulator is not an entirely fair comparison. Nevertheless, the SQP method can still be used for the comparison.

It is clear from the figure 26 below that the SQP optimizer constraint-handling showed by solid lines is better than the simulator constraint-handling showed by dashed lines. The simulator constraint-handling would find an optimal strategy based on the assumption that there is no constraint, so in other words the optimal strategy for unconstrained optimization, after which the constraints are introduced into the calculation. At this point, the simulator will just adjust the control by reducing it proportionally to the constraint violation (e.g. the excess of allowable field injection rate) using equation (27) and (28). By doing this, the constraints are obeyed but now the (control) strategy is no longer an optimal strategy. That is why this simulator constraint-handling results in a lower objective function value than the optimizer constraint-handling and is not an optimal strategy.

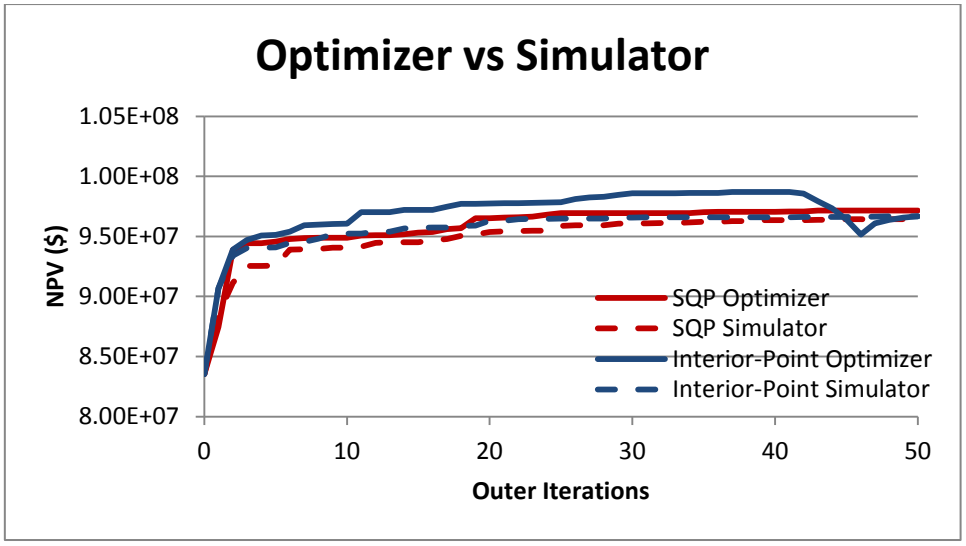


Figure 26. Input constraints with ensemble gradients, optimizer vs simulator

3.3.7 Bound Constraints with Regularization

From the results above, it is found that better gradients lead to higher objective function values. Because of this reason, some possible improvements that can be made in the ensemble gradients are interesting to be tested. The general gradient regularization option is available for the trust region optimizer to improve the quality of the gradients. To observe and find out the effects of these options on optimization process, some experiments were tested with the ensemble gradients. Since the trust region optimizer can only work with bound-constrained optimization, these gradient regularization experiments are only done for bound-constrained optimization. The results of the experiments are presented below.

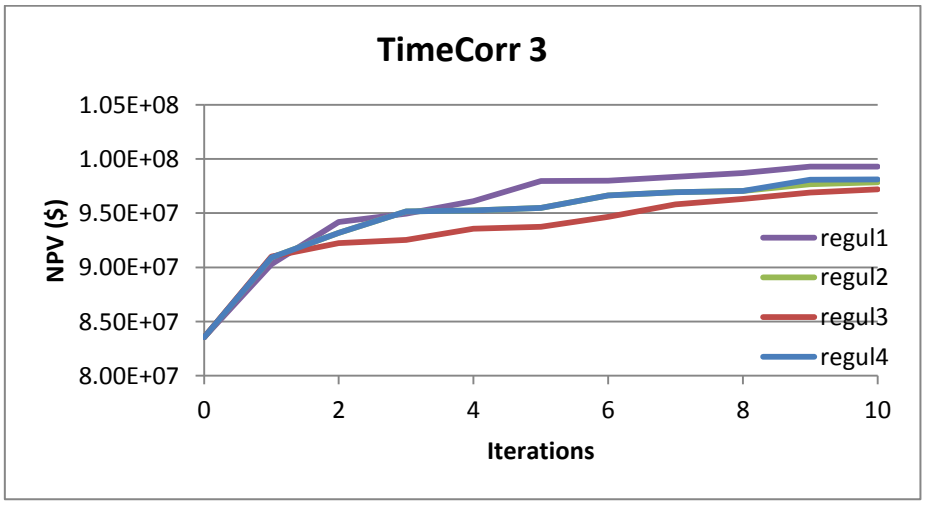


Figure 27. TimeCorr = 3 control time steps with sensitivity test on regularization

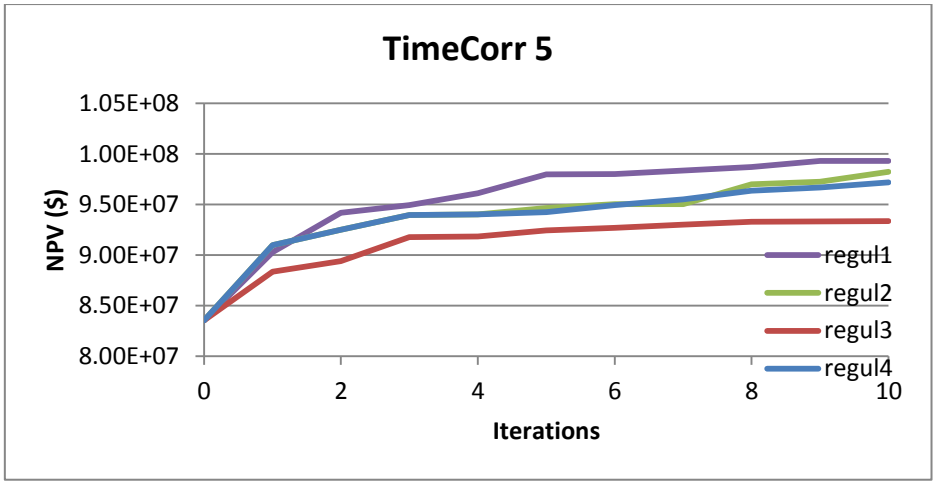


Figure 28. TimeCorr = 5 control time steps with sensitivity test on regularization

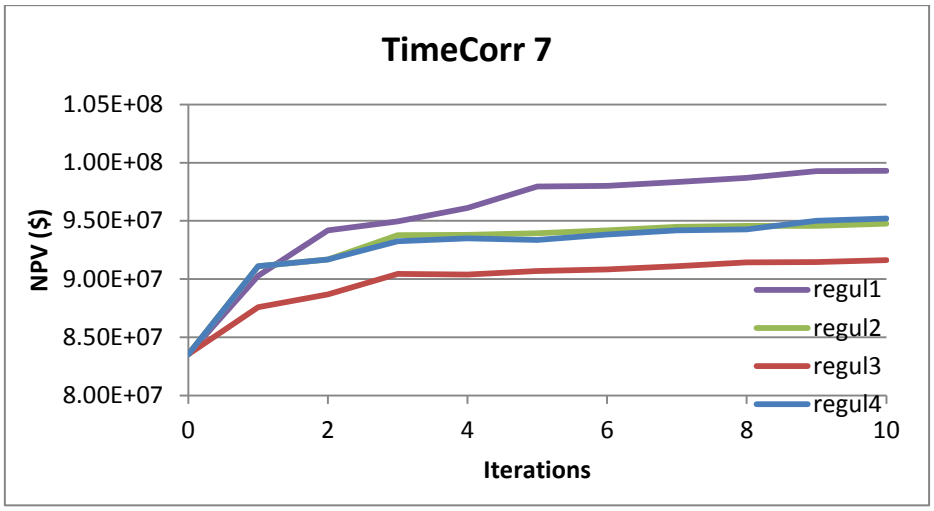


Figure 29. TimeCorr = 7 control time steps with sensitivity test on regularization

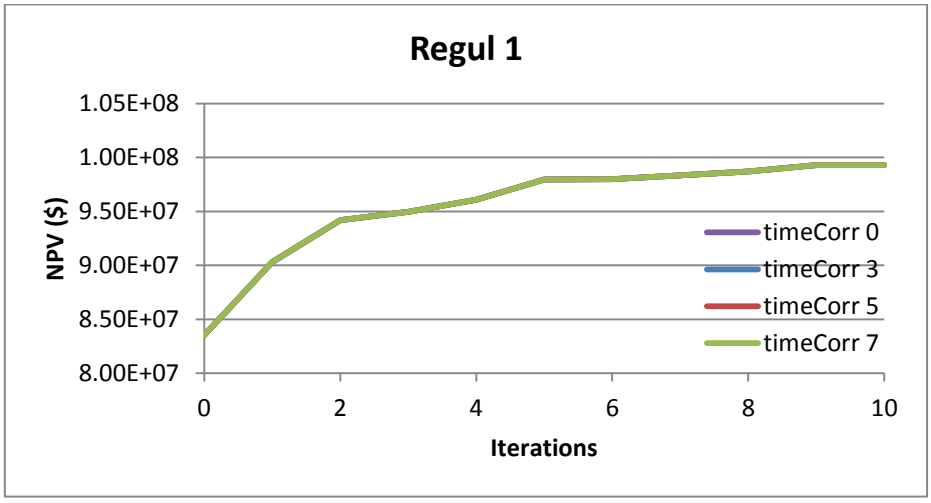


Figure 30. Regularization option 1 (no correlation is imposed on the control perturbation) with sensitivity test on time correlation

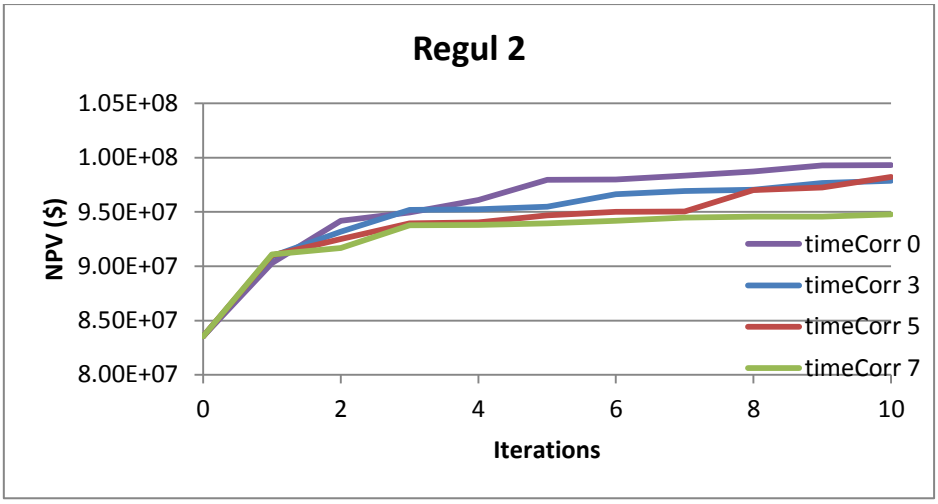


Figure 31. Regularization option 2 with sensitivity test on time correlation

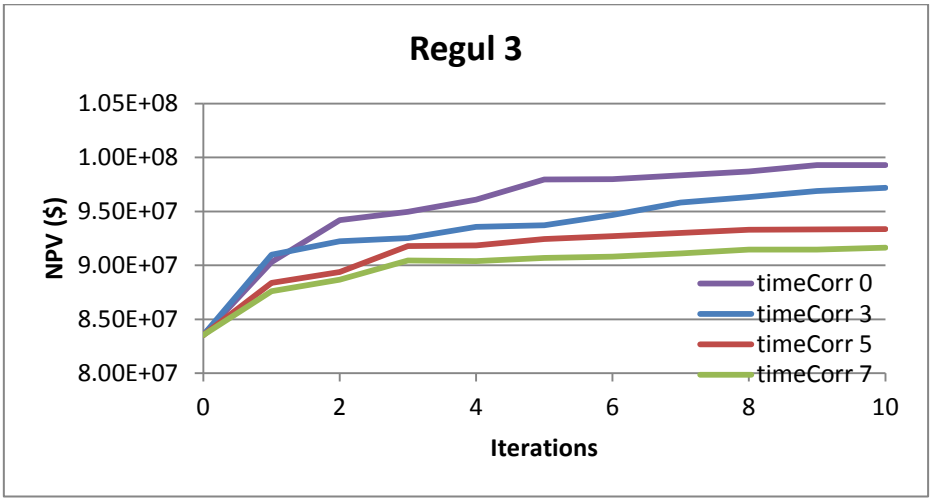


Figure 32. Regularization option 3 with sensitivity test on time correlation

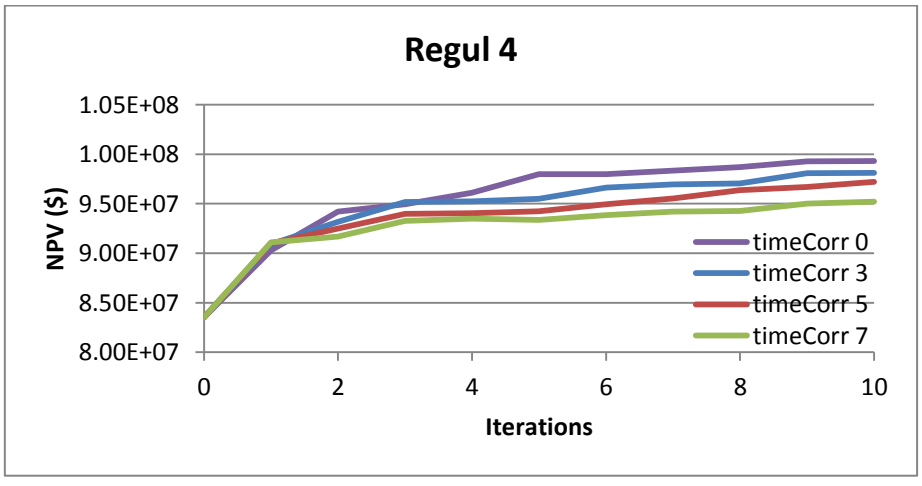


Figure 33. Regularization option 4 with sensitivity test on time correlation

In figures 27, 28, and 29 above, all the three plots have the same profile: regularization option 1 indicated by the purple-colored line gives the same values and also the highest NPV while regularization option 3 indicated by the red-colored line gives the lowest NPV for all different values of time correlation used. It is because regularization option 1 means the gradient of the objective function is not regularized which means all the controls perturbation are independent of each other and will not be affected by the neighbor controls. This will result in more freedom on control perturbation and thus this series of controls can achieve higher objective function values. The results of regularization option 1 gives the same values can also be seen in figure 30. Because the regularization is not active, it will just give the same results no matter the value of the time correlation is used.

Regularization option 4, on the other hand, is slightly different than regularization option 2 and 3. Regularization option 4 tries to smooth the gradient using Gaspari correlation function with some modification. Since the premultiplication on regularization option 4 is only done once and not twice like regularization option 3, and also the premultiplication itself is based on Gaspari correlation function, then it can be expected that the results are close to the results of regularization option 2. From figure 27, 28, and 29 it can be seen that the results between regularization option 2 and 4 for all time correlation values tested are very close to one another. Regularization option 3 that shows the lowest NPV for all cases is also as expected because the premultiplication calculation is done twice which will make the controls strongly correlated to one another. It means the control perturbations become more rigid and less flexible, and resulting in lower NPV than the other options.

From figure 31, 32, and 33, it can be seen that time correlation 0 gives the highest NPV values, just like the results from regularization option 1. It is because when time correlation 0 is chosen, no correlation between neighbor control variables is imposed, thus each control variable is independent variable and the control perturbations can be done in a more flexible way. Next to time correlation 0, time correlation 3 performs second the best and time correlation 7 performs the worst. This results are as expected and pretty straight forward because the bigger the value of the time correlation, the further the well controls are correlated one to another in time, (i.e. the perturbation of the well-controls in the future time-controls are affected by the perturbation of the well controls in the past or current time-controls) which leads to a more rigid and less flexible control and smoother controls strategy.

3.4 Conclusions

There are several conclusions that can be drawn up until now. For bound constraints, the trust region strategy outperforms the other three other optimizers in this case but unfortunately the trust region optimizer that is tested could not be used for input-constrained optimization. For optimizing the input-constrained problem, SQP is the better optimizer compared to Interior Point and Active Set in terms of the objective function values, the number of function evaluations, and the feasibility. Interior Point delivered higher NPV than SQP during early times but it was because the Interior Point was still in the infeasible region, which means the SQP can get inside the feasible region faster than

Interior Point. To move from the infeasible to the feasible region, the optimizer needs to update or change the controls even to the point where the objective function is sacrificed and decreased to find the feasible region. On the other hand, Active Set performed the worst in terms of the delivered NPV, the number of function evaluations needed, and the speed to move from the infeasible to the feasible region.

To obtain the approximate gradient, two methods were tested: finite difference gradients and ensemble gradients. The finite difference gradient method results in a higher NPV but at a significantly higher computational cost. It is less efficient than the ensemble gradient method because of the difference in the architecture on how to obtain the gradient although it has better accuracy in determining the approximate gradient.

Constraint handling can be done either by the optimizer or by the simulator. Leaving the constraints to be handled by the optimizer is the optimal way and delivers better results in terms of the NPV and also the iteration efficiency than by the simulator as shown in figure 26.

Correlation in the objective function gradient over multiple control time steps was tested as a regularization option. The stronger the smoothing process taken place (e.g. regularization option 3 applies the smoothing process twice), the stronger the correlation between the controls on different time steps. How strongly the gradient is correlated over time is also determined by the value of the time correlation. The bigger the time correlation input value, the further in time steps one well control will be affected by the other well control. It will produce less flexible strategy and thus lower the NPV but with smoother controls.

Chapter 4: Output-Constrained Optimization

4.1 Model Description

Input constraints can be evaluated once the proposed controls are known, whereas output constraints can only be evaluated by running a simulation with the proposed controls. For an example, in an input-constrained optimization problem the control variables could be the injection rates and the constraint the field injection rate, while in an output-constrained optimization problem the control variables could be the injector ICV settings and the constraint is the liquid production rate.

The same reservoir model as in the previous experiments for input-constrained optimization problems is used. The field liquid production rate is selected as the constraint and is limited at 300 m^3/day while each injector has a maximum ICV opening of 1 and a minimum ICV opening of 0. An initial ICV opening used is 0.5 with a perturbation size of 0.05. The length of the life-cycle, the number of simulation time steps and control time steps, and the reservoir simulator are kept the same. More detailed information about the L-shaped sealing fault reservoir model can be found in the previous subchapter 3.1.

4.2 Results and Discussion

4.2.1 Different Methods for Output Constraints Optimization

Matlab *fmincon* provides three methods to solve nonlinear optimization problems: the Active Set, Interior Point, and SQP methods. During the input-constrained optimization test cases, it was found that the Active Set method is the worst method of the three in terms of the ability to find the feasible region, the final objective function value, and the number of function evaluations to reach the optimum solution.

During initial simulations run for output-constrained optimization with the three methods, again the Active Set method showed the worst performance, especially in terms of stability of the objective function as shown in figure 34 below. For this reason, it was decided to leave this method behind, leaving only two methods to be considered: SQP and Interior Point.

After performing several preliminary experiments, it was found that Interior Point always evaluates both the gradients of the objective function and the constraints while doing the backtracking evaluation. On the other hand, SQP uses a different algorithm where it only evaluates the gradients for both the objective function and the function evaluations when it finds the acceptable solution and then updating the controls. In other words, SQP will not evaluate the gradients when it only does backtracking evaluations. This difference on how both optimizers operate and handle backtracking evaluations makes the SQP method more efficient in terms of the total number of function evaluations needed, especially in a highly non-linear problem where a lot of backtracking

evaluations will be needed to ensure that the solutions remain in the feasible region while optimizing the objective function at the same time.

In the following experiments, the number of outer iterations will be limited to a maximum of 30. This is to make sure that the experiments are efficient: enough iterations to get significant improvement in the objective function while avoiding an excessive number of iterations but an insignificant objective function value improvement at the same time. The control tolerance $TolX$ is also set to be 0 to make sure the optimization process will not stop prematurely. Control tolerance $TolX$ is a minimum value for the control difference between current and previous iterations to keep the iteration running. If the difference between two consecutive iterations is smaller than the control tolerance $TolX$, the iteration process will stop.

To test the robustness of the experiment, the experiments will be started from the infeasible region. This way, the optimization process will become more challenging because now the optimizer will not only have to find the optimal strategy to get the maximum objective function, but will also have to move from the infeasible to the feasible region to maintain the constraint restrictions. The initial ICV control used is 0.5 opening for all injector wells. The production profile from this constant half-opened ICV strategy for the whole reservoir life can be seen in figure 35.

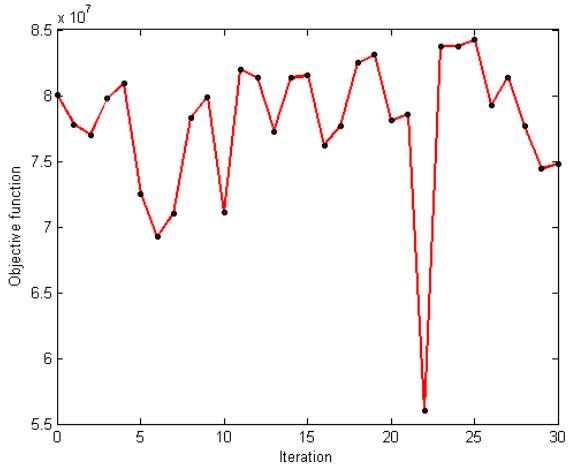


Figure 34. The objective function of output-constrained optimization with Active Set

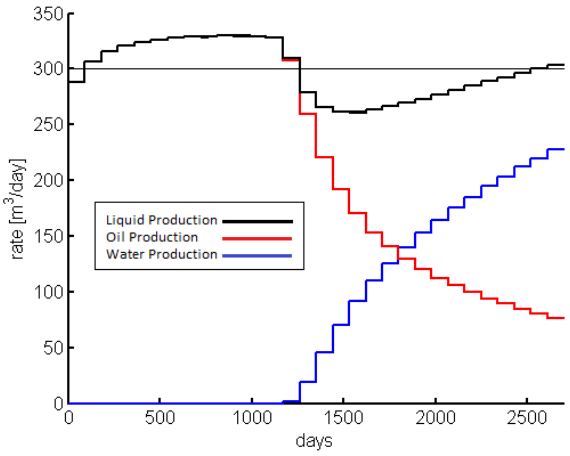


Figure 35. Production rate profile for ICV constant 0.5

It can be seen from figure 35, that if constant a half-opened ICV strategy is used, it will violate the constraint of maximum liquid production at 300 m^3/day. Thus, this strategy is an infeasible strategy.

Based on the initial ICV input, the reservoir is then optimized with both Interior Point and SQP optimizers as it can be seen in figure 36 below. After 30 iterations, the Interior Point gave an NPV value of $\$\,8.55 \times 10^7$ while the SQP gave $\$\,8.56 \times 10^7$. The difference between these two final results may seem not significant, but the processes to reach these NPV values are different. During the early iterations, SQP performed better than Interior Point, with the widest gap between the two optimizers equal to $\$\,0.115 \times 10^7$ and this happened at the 14th iteration. Based on this result, it can be said that in this case SQP has a higher convergence speed compared to Interior Point.

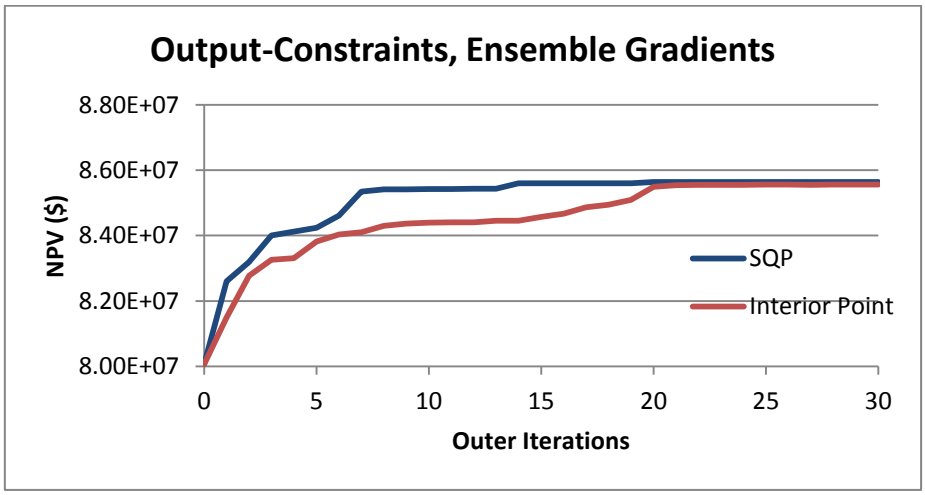


Figure 36. Output constraints with ensemble gradients, Interior Point vs SQP

The liquid production rate profile of both the Interior Point and SQP can be seen in figure 37 and 38 below.

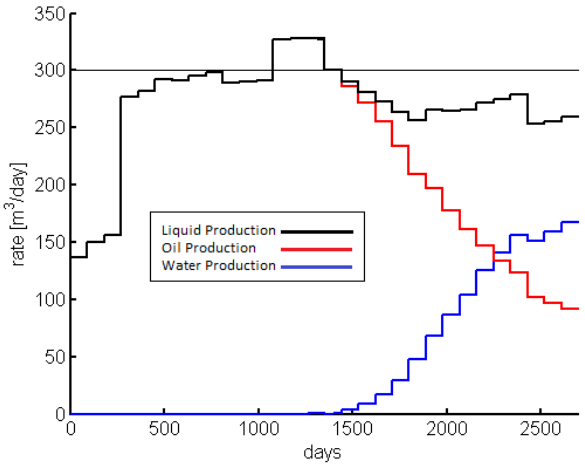


Figure 37. Production rate profile for Interior Point

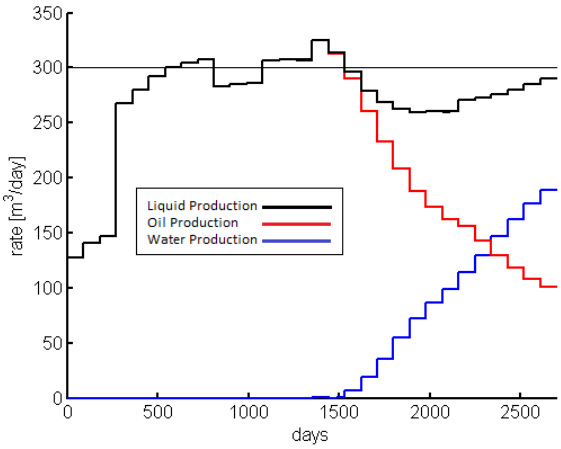


Figure 38. Production rate profile for SQP

As can be observed from figures 37 and 38, both Interior Point and SQP violate the constraints. It turned out that the scaling of the constraints is very important, and this will be further investigated in the next section.

4.2.2 Constraint Scaling

Several experiments were conducted to gain a better insight and also to solve the constraint violation in figures 37 and 38 above. One experiment that shows significant improvement on constraint adherence is the implementation of constraint scaling. Constraint scaling is basically multiplying the constraint vector with a predetermined value to change the order of magnitude of the constraint vector: it can be scaled up or scaled down.

The reason behind the constraint scaling implementation is to scale up the constraints' order of magnitude thus making the constraint violation become more significant. In this problem, the objective function has an order of magnitude 10^{7} while the constraint vector has an order of magnitude 10^{-4}. By up-scaling the constraints, not only the constraint violation will be penalized more, but also will the objective function and the constraints (and also the constraint gradients) become of the same order of magnitude.

The current constraint violation's order of magnitude is a relatively small number and close to the constraint violation tolerance $TolCon$ of 10^{-6}. The constraint violation tolerance $TolCon$ is a maximum value up to which constraint violation is still allowed. If the constraint violation is bigger than the $TolCon$ value, the optimizer will see the solution as an infeasible solution and therefore a proper handling to the current controls needs to be taken. Because of the small value of this $TolCon$, it may lead to false perception of the optimizer about the importance of the constraint adherence.

By multiplying the constraint vector and also the constraint gradients with a multiplier of 10^{11}, the two ideas above can be addressed at the same time: making the objective function and the

constraints to the same level of order of magnitude and also penalizing the constraint violation more to obtain stronger constraint adherence. Figure 39 and figure 40 show the production rate profile of Interior Point and SQP respectively and figure 41 shows the NPV difference after the constraint scaling implementation (i.e. after the constraints being scaled up).

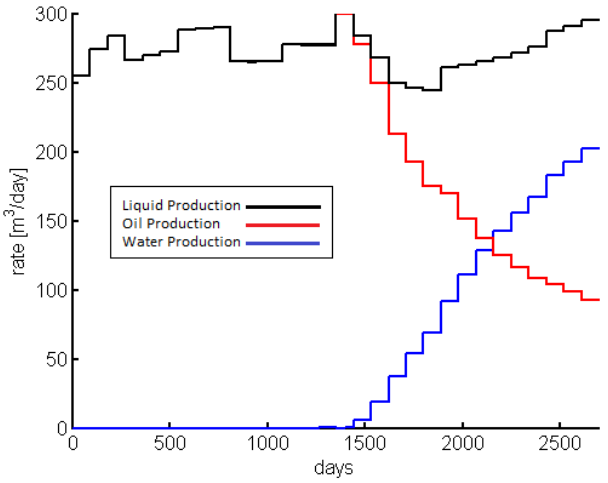


Figure 39. Production rate profile of Interior Point with constraint scaling

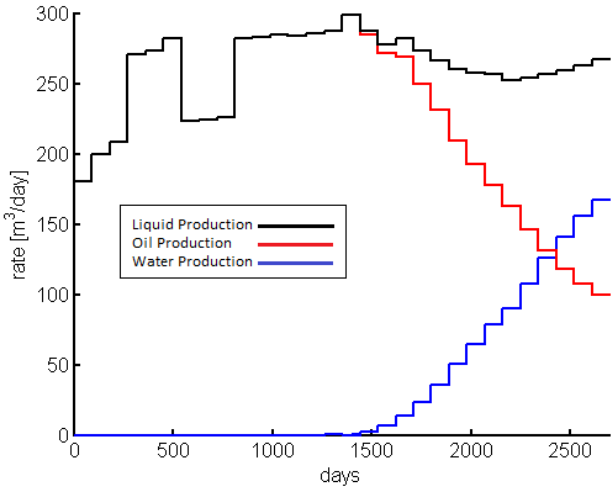


Figure 40. Production rate profile of SQP with constraint scaling

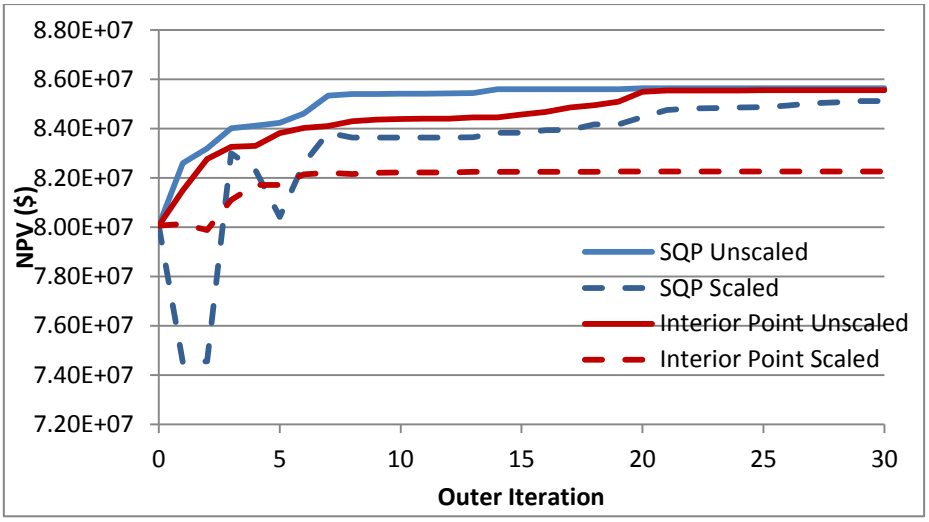


Figure 41. NPV Difference between SQP and Interior Point in outer iterations

From figure 39 and 40 above, it can be seen that now both optimizers obey the constraints strictly and better than before. This constraint adherence comes at the cost of lower NPV values as can be seen in figure 41, which is understandable because the additional NPV contribution comes from the production rate that violates the constraints. From figure 41, it can also be seen that the SQP again gives higher NPV values compared to the Interior Point. An interesting observation is that there are two big drops in the blue dashed line (indicating SQP) after constraint scaling. The first drop happened right after initialization and the second drop happened after the third iteration. After evaluation of the diagnostic information, it was concluded that the sudden drop was caused by the controls update done by the optimizer to avoid high constraint violation.

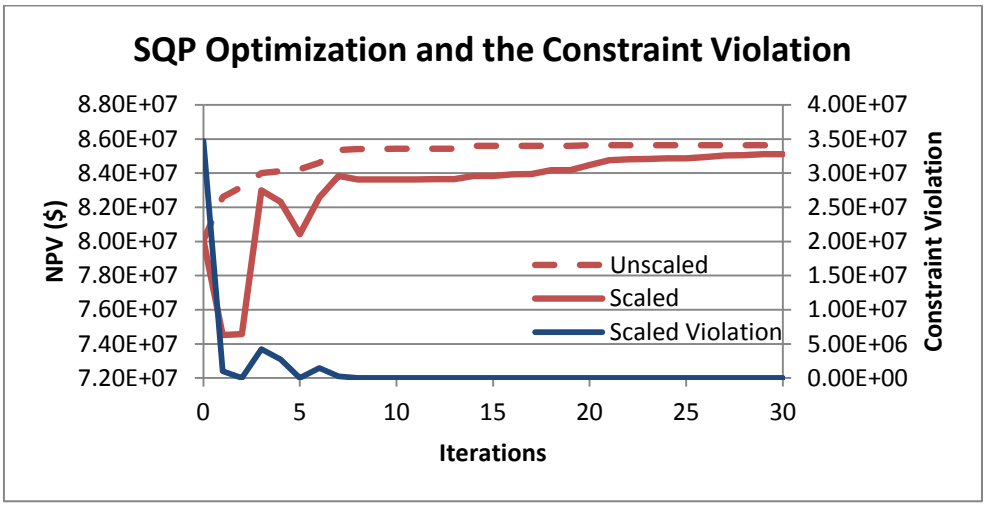


Figure 42. SQP optimization and the constraint violation

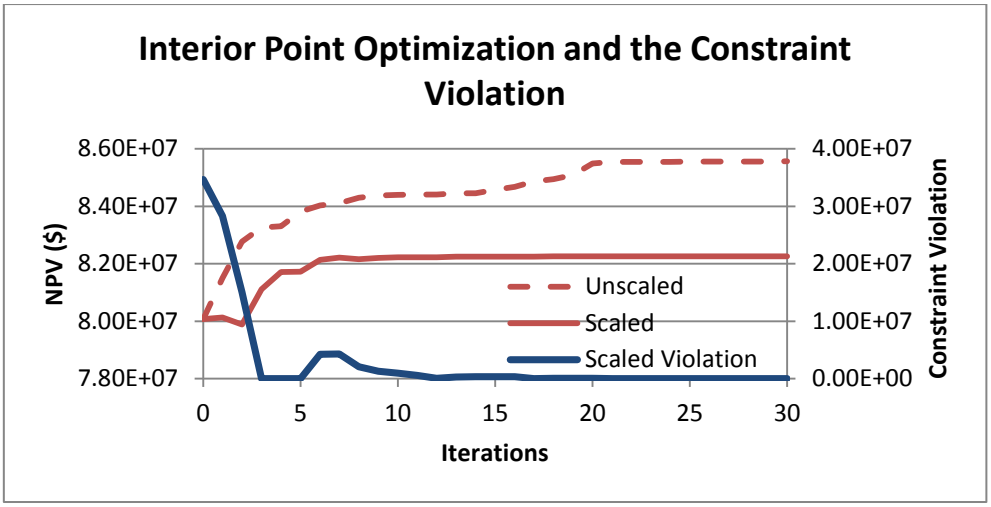


Figure 43. Interior Point optimization and the constraint violation

From figures 42 and 43, it can be seen that right from the beginning (i.e. iteration 0), the constraint violation, indicated by the blue line, already showed high a value thanks to the constraint scaling implementation. This high constraint violation value leads to an attempt by the optimizers to update the strategy so that the constraint violation value is reduced as low as possible to move from the infeasible to the feasible region. The result was a sudden drop in the NPV, indicated by the solid red line. The SQP can obtain constraint violation 0 after 8 iterations while the Interior Point can only obtain constraint violation 0 after 23 iterations. It can be said that the SQP is faster than Interior Point in adhering the constraints. From figures 42 and 43, the constraint violation for the unscaled case is not visible because it still has a magnitude of 10^{-4} and thus very small compared to the constraint violation of the scaled case.

Because of the difference in nature of how the two optimizers execute the functions, where the Interior Point always evaluates the gradients of both the objective function and the constraints, it is needed to compare the objective function of figure 41 versus function evaluations instead of the outer iterations. Figure 44 shows the difference.

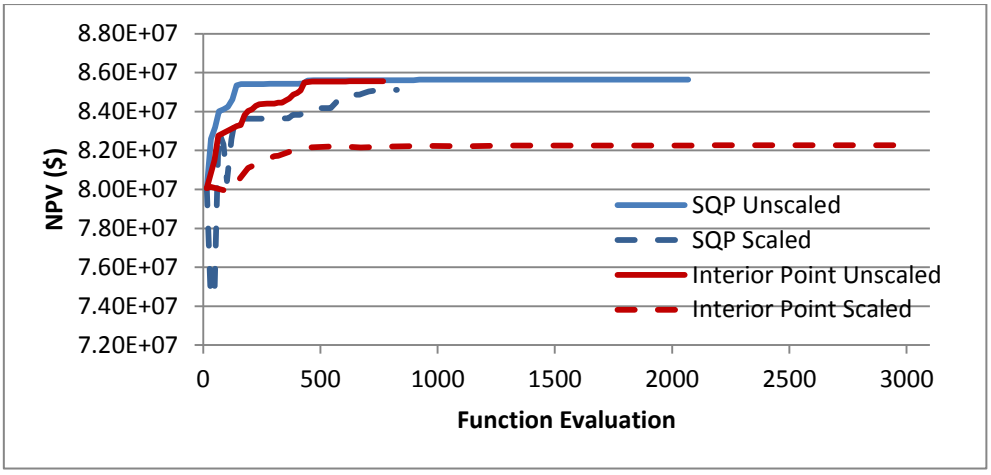


Figure 44. NPV Difference between SQP and Interior Point in function evaluations

From figure 44, by comparing the blue and red dash lines for SQP and Interior Point respectively, it is clear that even though both optimizers use the same number of outer iterations, the numbers of function evaluations executed are very different. The SQP only requires 706 function evaluations while the Interior Point needed 3040. Adding the fact that the NPV obtained by SQP with less function evaluations is still higher than the Interior Point's, and also previous results that SQP performs better than Interior Point, it can be said that in this problem the Interior Point is less efficient than SQP. Therefore, from now on the experiments will focus on SQP optimizer only.

4.2.3 Different Ensemble Sizes

The idea of using the ensemble gradient method is to obtain approximate gradient information with less computation time needed than for evaluating the finite difference gradient. The number of simulations needed for each iteration depends on the number of ensemble members: the more ensemble members the more computation time needed. The idea of using many ensemble members is to obtain a better-quality gradient approximation. Therefore, the ideal ensemble size is less than the number of controls, small enough to minimize the computational time but sufficient to get a good-quality gradient approximation and thus a high increment of the objective function value during the optimization process.

In this experiment, there are 40 well controls which equals to 41 simulations needed for the finite difference to obtain a gradient. As comparisons, these finite difference gradients will be compared with the ensemble gradients with 15 and 50 ensemble members, both with and without constraint scaling.

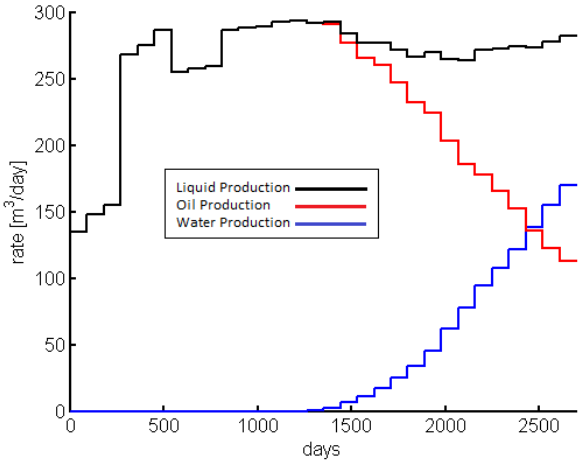


Figure 45. Production rate profile with 50 ensemble members, unscaled

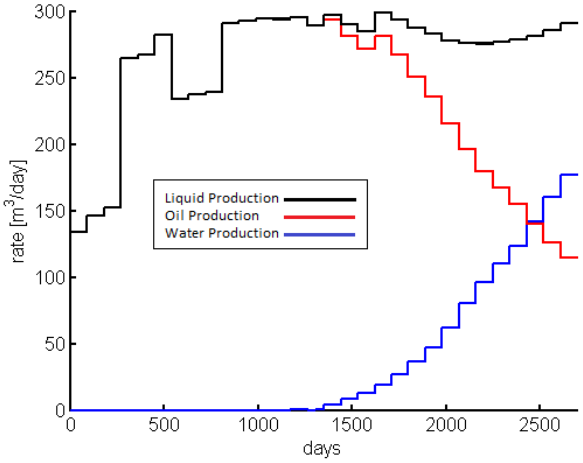


Figure 46. Production rate profile with finite difference

From figure 45 and 46 above, it can be seen that with 50 ensemble members and finite difference, the constraint adherence can be obtained without using the constraint scaling feature. This adherence could not be obtained using only 15 ensemble sized as shown in figure 37 and 38 above, probably because of the difference in gradient quality. With only 15 ensemble members, it has some trouble in obeying all the constraints.

From figure 47 and 48 below, it can be seen that after 30 outer iterations, the final objective function values from 15 and 50 ensemble members are almost the same with the 50 ensemble members producing a slightly higher value than the 15 ensemble members. Based on intermediate results of the optimization process, it appears that the ensemble gradients with 15 members produce a faster increment during the early iterations while the ensemble gradients with 50 members has better constraint adherence. One interesting observation is that the finite difference gradients perform better than the ensemble gradients with 50 members. It is understandable that the finite difference performs better than the ensemble gradients with 15 members because it is

known from the beginning that the finite difference gradients require more simulations therefore it is also expected to perform better.

The situation is different with the ensemble gradients with 50 members. With 50 ensemble members, now it requires more simulations than the finite difference. By having more simulations, it is expected to have better result, but from figure 47 and 48 below, it can be seen that the finite difference still performs better than the ensemble gradients with 50 ensemble members. As already explained in the previous subchapter 3.3.5 above, this might be caused by the way the finite difference method determines the gradients. By changing the controls one-by-one, even though it is not very efficient, but it is more systematic and structured in obtaining the gradients. Based on these results, it is concluded that it is not efficient to use ensemble gradients to optimize a problem using more ensemble members than the number of the controls the problem has. It is suggested to either choose the ensemble gradients with less ensemble members for the sake of efficiency or choose the finite difference method for a better result instead.

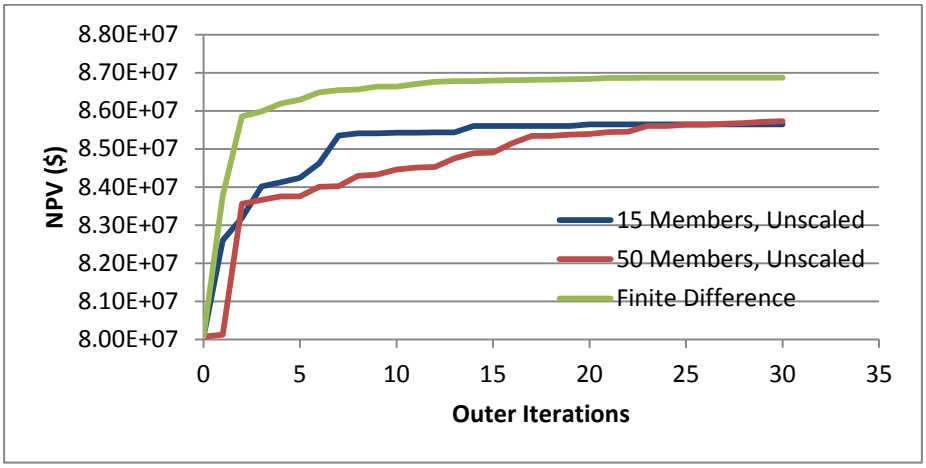


Figure 47. NPV vs Outer iterations with different ensemble members and unscaled

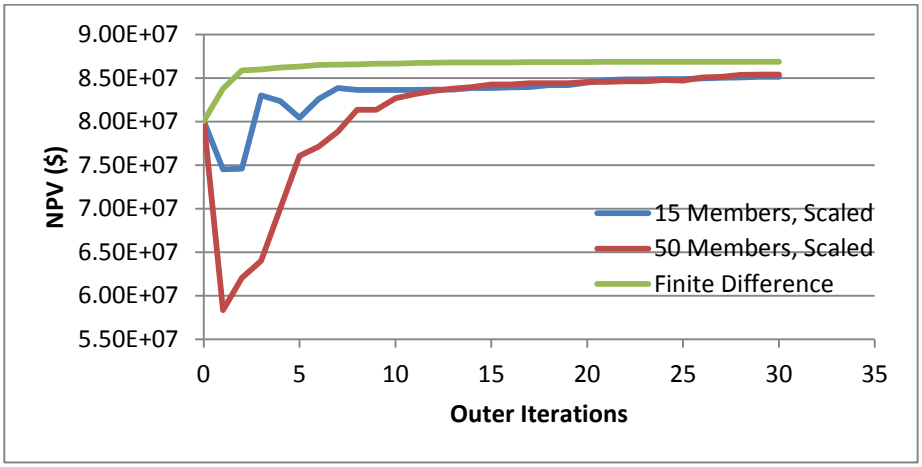


Figure 48. NPV vs Outer iterations with different ensemble members and scaled

From figure 48, it can be seen the same effect occurred with the constraint scaling implementation as already discussed earlier. Both the ensemble gradients with 15 and 50 ensemble members updated their controls from the first iteration such that they are in the feasible region now, thus resulting in NPV drops.

4.2.4 Different Number of Control Time Steps and Gradient Regularization

In this experiment, how the number of control time steps plays a role in the optimization will be investigated. The number of the control time steps was changed from 10 to 30 which mean now there are 120 controls. To provide a better adherence to the feasibility, constraint scaling was used for the experiments. Below is the plot showing the difference between two different numbers of control time steps in the objective function.

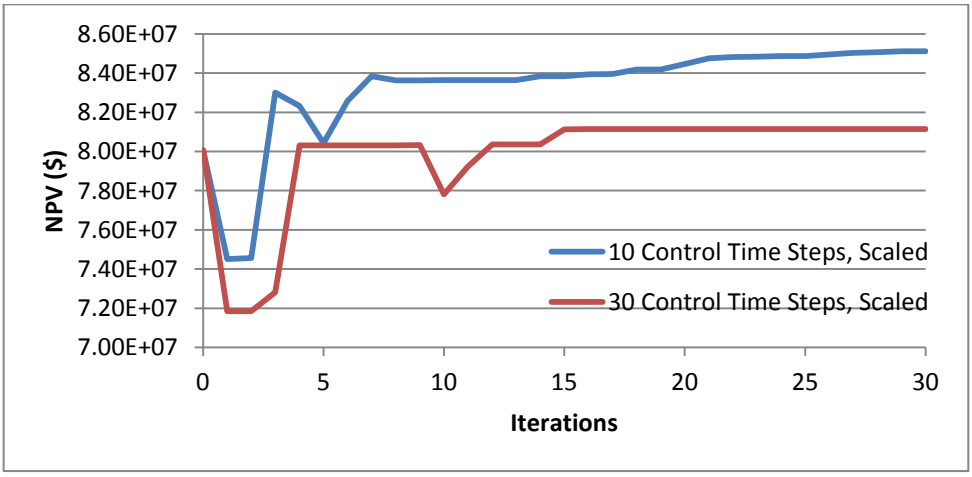


Figure 49. 15 vs 30 Control time steps on objective functione

From figure 49, it can be seen that the one with 10 control time steps has a higher NPV than the one with 30 control time steps. This may initially appear counterintuitive because more controls means more degrees of freedom to optimize the strategy. The initial hypothesis is that since the gradients that were used are approximate gradients and not analytic gradients, having more degrees of freedom (e.g. more controls) does not help getting higher NPV. Instead, the NPV became lower because the gradients were not accurate enough and using more controls means being more prone to errors. This inaccuracy can be seen in figure 50 where the controls fluctuate sharply, probably indicating low-accuracy gradients.

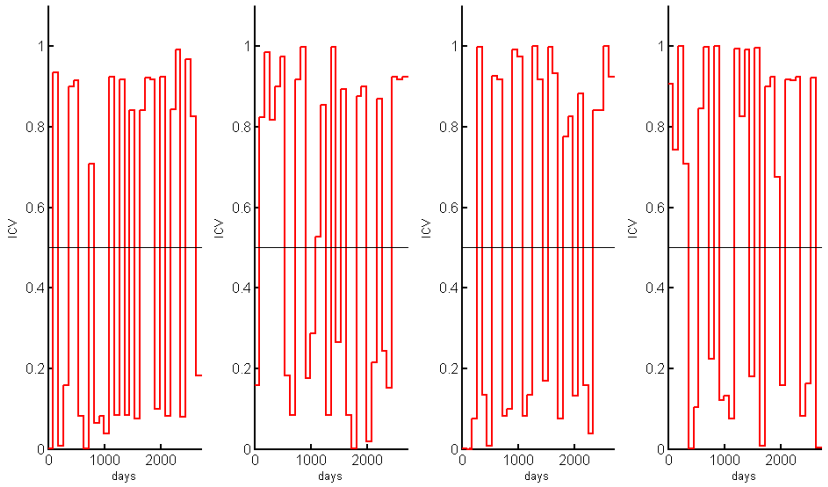


Figure 50. Controls on ICV with 30 control time steps

One way to mitigate the gradient inaccuracy is by implementing gradient regularization. As already explained in the previous chapter, by implementing the gradient regularization, the controls are correlated in time and the gradients are smoothed, thus removing the sharp changes it previously had in the controls. Figure 51 below shows the solution for ICV strategy with 30 control time steps that was regularized with regularization option 4 and time correlation 4.

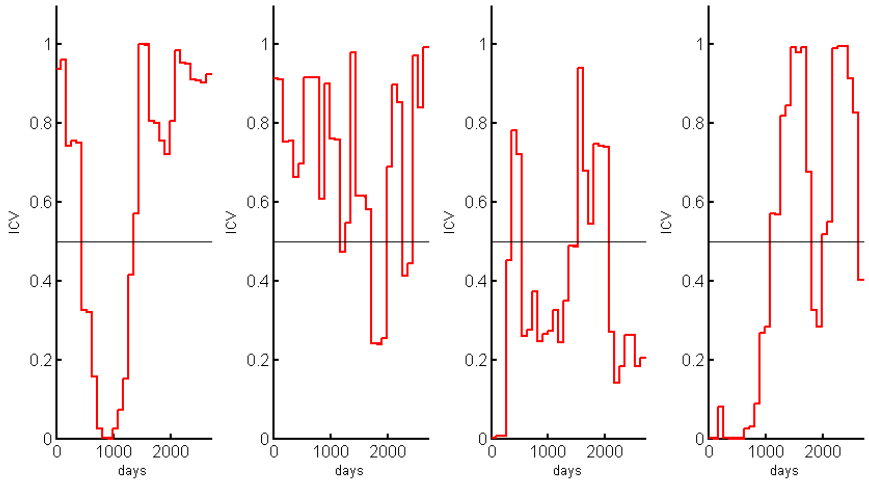


Figure 51. Controls on ICV with 30 control time steps and regularized

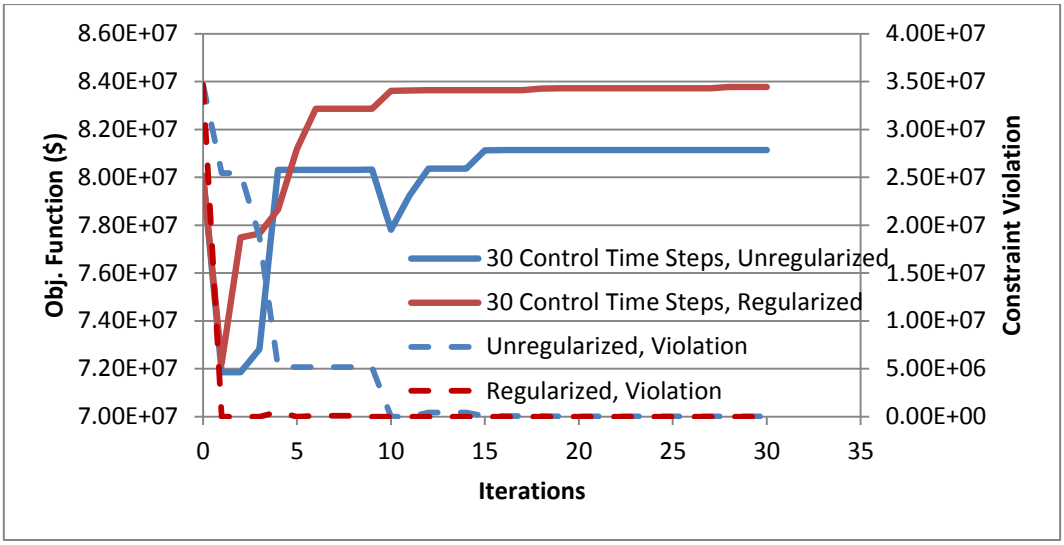


Figure 52. Objective function and constraint violation with 30 control time steps

In figure 52 above, it can be seen that the implementation of the gradient regularization significantly helps improving the optimization process, not only in terms of the NPV but also the speed to move from infeasible to feasible region. With gradient regularization, the NPV is constantly higher than the one without regularization. The constraint violation also shows no violation after the first iteration while for the one without regularization the constraint violation shows no violation only after 10 iterations.

These results are different than the results obtained in the input-constrained optimization experiments whereas the implementation of gradient regularization made the objective function lower than the one without. This difference might be caused by the difference in the number of the controls. In the previous experiments, the regularization implementation was tested for 10 control time steps while in these experiments it was tested for 30 control time steps, three times more controls than in the input-constrained experiments. In the previous experiments, since there are 10 control time steps, probably the right explanation is that the gradient regularization implementation is not needed because the quality of the ensemble gradients is still good enough and gradient regularization instead will hold back and restrict the gradient's freedom. On the contrary, with 30 control time steps, the quality of the ensemble gradients is not good enough and having too many controls with inaccurate gradients will only make the optimization process suboptimal.

4.2.5 Time Localization in Constraint Gradients

Because of the way the ensemble is perturbed, the constraint gradients will always have non-zero values for all time steps. Based on causalities principle, this is not the correct way to express the constraint gradients. The idea of the time localization is that any well control that is changed at a time step should only affect the current and the future time step constraints and not the past time

step because it has already happened. It is believed that with time localization, the gradient will become more accurate.

Based on the result shown in figure 53 below, the time localization gives slightly higher NPV compared to the one without after 7 iterations while in the earlier iterations the one without time localization give higher NPV. This result is as expected because by implementing the time localization, now a change in time step will not affect the past time step, which in a sense will reduce the flexibility of the optimization process. That is why the optimization with time localization implementation is slower in reaching its optimum solution. It is important to remember that having more accurate gradient does not always mean better improvement in the optimization process. Further research about time localization is still needed to give a better understanding about this implementation in gradient-based optimization.

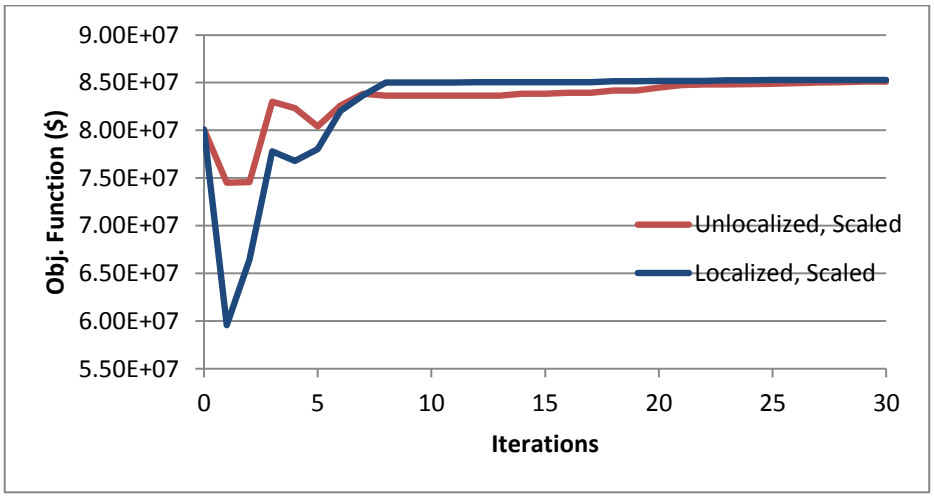


Figure 53. Time localization effects on objective function

4.3 Conclusions

There are several conclusions that can be drawn from this output-constrained optimization experiments. The Interior Point evaluates both objective function gradient and constraint gradient when executes the backtracking evaluation while the SQP only evaluates the constraint. This difference makes the SQP more efficient in terms of the total number of function evaluations needed.

Some experiments were done in order to have a better understanding on how to handle the constraints. One experiment that shows significant improvement on constraint adherence is the constraint scaling. When using the constraint scaling, the SQP reaches higher objective function value and also get inside the feasible region faster than the Interior Point. This result is consistent with the previous results that showed the SQP is the better optimizer in this experiment.

Other change that can be done to improve the constraint adherence is by changing the number of ensemble size. When the ensemble size is increased from 15 to 50, the constraint adherence is also improving. After 30 outer iterations, no constraint violation is seen. Finite difference also shows the same result where no constraint violation is seen.

In the regularization experiment for the output-constrained optimization, the number of control time steps is increased from 10 to 30. By having more control, it is more difficult to find the optimum strategy because the gradient that is used is not an exact gradient. The result is a lower objective function value compared with the 10 control time steps. After using the regularization option, the solution strategy becomes smoother, the objective function value becomes higher, and the constraint adherence also becomes better.

Chapter 5: Conclusions and Recommendation

5.1 Conclusions

1. This research has demonstrated that it is possible, in principle, to efficiently solve output-constrained optimization problems by an ensemble-gradient approach.
2. Conclusions that are related to the optimizer performance may in some cases be related to the specifics of the implementation in fmincon, the details of which are not accessible.
3. For input-constrained optimization problems, SQP is the better optimizer compared to Interior Point and Active Set in terms of the objective function value, the number of function evaluations, and the feasibility. SQP can get inside the feasible region faster than Interior Point while Active Set performed the worst.
4. Even though the constraint handling can be done either by the optimizer or by the simulator, in general it is better to let the optimizer handle the constraint.
5. The finite difference gradient has better accuracy than the ensemble gradient in determining the approximate gradient and gives higher objective function values for the same or more number of function evaluations.
6. Increasing the number of ensemble members will increase the objective function value by improving the quality of the gradient approximation at the cost of computation time. This better-quality gradient can also help adhering to the constraints.
7. The ideal ensemble size is smaller than the number of controls, small enough to minimize the computational time but sufficient to obtain high-quality gradient approximations and thus high increments of the objective function value during the optimization process.
8. Constraint scaling can help the constraint adherence by penalizing the constraint violation more. In this output-constrained optimization case, SQP is faster than Interior Point in adhering to the constraints.
9. Having too many variables may lead to a lower objective function values due to inaccuracies caused by the approximate gradients used. Regularization (smoothing) implementation can help improving the objective function values.
10. Having too strong regularization process is also undesirable because it will make the solution strategy becomes too smooth and lose some details.

5.2 Recommendation

Even though a lot of simulation experiments have been done, there are still other opportunities and features to be explored and optimized for this field. Some topics for future works to consider in this constrained optimization topic are constraint-lumping, multi-objective constrained optimization, CMA (Covariance Matrix Adaptation) EnOpt for constrained optimization, and adding spatial variables (e.g. well location) as the controls. It is also important to test the optimization program with a more complex but still commonly used reservoir model, e.g. the Brugge reservoir model, as a benchmark study for flooding optimization. By doing so, the results and conclusions obtained with

the L-shaped fault reservoir model can be confirmed to be also applicable to more complex reservoir models.

REFERENCES

Asadollahi, M., Naevdal, G., Markovinovic, R., Shafieirad, A. 2009. A Workflow for Efficient Initialization of Local-Search Iterative Methods for Waterflooding Optimization. Paper IPTC 13994 presented at IPTC Conference, 7-9 December 2009, Doha, Qatar. DOI: 10.2523/13994-MS.

Bartlett, R.A., Wachter, A., and Biegler, L.T. 2000. Active Set vs. Interior Point Strategies for Model Predictive Control. American Control Conference, 28-30 June 2000, Chicago, Illinois. DOI: 10.1109/ACC.2000.877018.

Brouwer, D.R., Naevdal, G., Jansen, J.D., Vefring, E. and van Kruijsdijk, C.P.J.W. 2004: Improved reservoir management through optimal control and continuous model updating. Paper SPE 90149 presented at the SPE Annual Technical Conference and Exhibition, Houston, Texas, USA, 26-29 September. DOI: 10.2118/90149-MS.

Chaudri, M.M., Phale, H.A., Liu, N., and Oliver, D.S. 2009. An Improved Approach for Ensemble-based Production Optimization. Paper SPE 121305 presented at SPE Western Regional Meeting, 24-26 March 2009, San Jose, California. DOI: 10.2118/121305-MS

Chen, C. 2011. Adjoint-Gradient-based Production Optimization with the Augmented Lagrangian Method. PhD Dissertation, University of Tulsa, USA.

Chen, Y. 2008. Ensemble-Based Closed-loop Production Optimization. PhD Dissertation, University of Oklahoma, USA.

Chen, C., Li, G., and Reynolds, A.C. 2011. Robust Constrained Optimization of Short and Long-term NPV for Closed-loop Reservoir Management. Paper SPE 141314 presented at SPE Reservoir Simulation Symposium, 21-23 February 2011, The Woodlands, Texas, USA.

Chen, Y., Oliver, D.S., and Zhang D. 2009. Efficient Ensemble-Based Closed-loop Production Optimization. SPEJ 14 (4): 634-645. SPE112873-PA. DOI: 10.2118/112873-PA.

Chen, Y. and Oliver, D.S. 2009. Localization of Ensemble-based Control Setting Updates for Production Optimization. Paper SPE 125042 presented at SPE Annual Technical Conference and Exhibition, 4-7 October 2009, New Orleans, Louisiana, USA. DOI: 10.2118/125042-MS.

Dehdari, V. and Oliver, D.S. 2011. Sequential Quadratic Programming (SQP) for Solving Constrained Production Optimization – Case Study from Brugge Field. Paper SPE 141589 presented at SPE Reservoir Simulation Symposium, 21-23 February 2011, The Woodlands, Texas, USA.

Egberts, P. and Leeuwenburgh, O. 2011. Note on Ensemble-based Life-cycle Optimization. TNO Internal Note.

Fonseca, R.M. 2011. Robust Ensemble-based Multi-Objective Production Optimization: Application to Smart Wells. MSc Thesis Report. Delft University of Technology, The Netherlands.

Fonseca, R.M., Leeuwenburgh, O., Van den Hof, P.M.J., Jansen, J.D. 2013. Improving the Ensemble Optimization Method through Covariance Matrix Adaptation (CMA-EnOpt). Paper SPE 163657 presented at SPE Reservoir Simulation Symposium, 18-20 February 2013, The Woodlands, Texas, USA.

Freund, R.M. 2004. Penalty and Barrier Methods for Constrained Optimization. Lecture Notes. Massachusetts Institute of Technology, USA.

Jansen, J.D., 2012. Advance Reservoir Simulation: System Theory for Reservoir Management. Lecture Notes AES 1490. Delft University of Technology, The Netherlands.

Jansen, J.D., Brouwer, D.R., Nævdal, G. and van Kruijsdijk, C.P.J.W. 2005: Closed-loop reservoir management. First Break, January, 23, 43-48.

Jansen, J.D., Bosgra, O.H. and van den Hof, P.M.J. 2008: Model-based control of multiphase flow in subsurface oil reservoirs. Journal of Process Control 18, 846-855. DOI: 10.1016/j.jprocont.2008.06.011.

Jansen, J.D., Douma, S.G., Brouwer, D.R., Van den Hof, P.M.J., Bosgra, O.H. and Heemink, A.W. 2009: Closed-loop reservoir management. Paper SPE 119098 presented at the SPE Reservoir Simulation Symposium, The Woodlands, USA, 2-4 February. DOI: 10.2118/119098-MS.

Kraaijevanger, J.F.B.M., Egberts, P.J.P., and Valstar, J.R. 2007. Optimal Waterflood Design Using the Adjoint Method. Paper SPE 105764 presented at SPE Reservoir Simulation Symposium, 26-28 February 2007, Houston, Texas, USA.

Leeuwenburgh, O., Egberts, P.J.P., and Abbink O.A. 2010. Ensemble Methods for Reservoir Life-cycle Optimization and Well Placement. Paper SPE 136916 presented at SPE/DGS Saudi Arabia Section Technical Symposium and Exhibition, 4-7 April 2010, Al-Khobar, Saudi Arabia. DOI: 10.2118/136916-MS.

Leeuwenburgh, O., Egberts, P.J.P., Chitu, A.G. 2013. Life-cycle Optimization using Stochastic Gradients. Technical Reference, TNO Internal Note.

Luenberger, D.G. 2003. Linear and Non-linear Programming (2nd ed.), Dordrecht, the Netherlands.

Naevdal, G., Brouwer, D.R. and Jansen, J.D. 2006: Waterflooding using closed-loop control. Computational Geosciences 10 (1) 37-60. DOI: 10.1007/s10596-005-9010-6.

Nocedal, J., Wright, S.J. 2006. Numerical Optimization (2nd ed.), Berlin, New York, USA.

Sarma, P., Chen, W.H., Durlofsky, L.J., Aziz, K. 2008. Production Optimization With Adjoint Models Under Nonlinear Control-State Path Inequality Constraints. SPEJ 11 (2): 326-339. SPE99959-PA. DOI: 10.2118/99959-PA.

Schittkowski, K. Optimization in Industrial Engineering: SQP Methods and Applications. Computer Science Lecture Notes. University of Bayreuth, Germany.

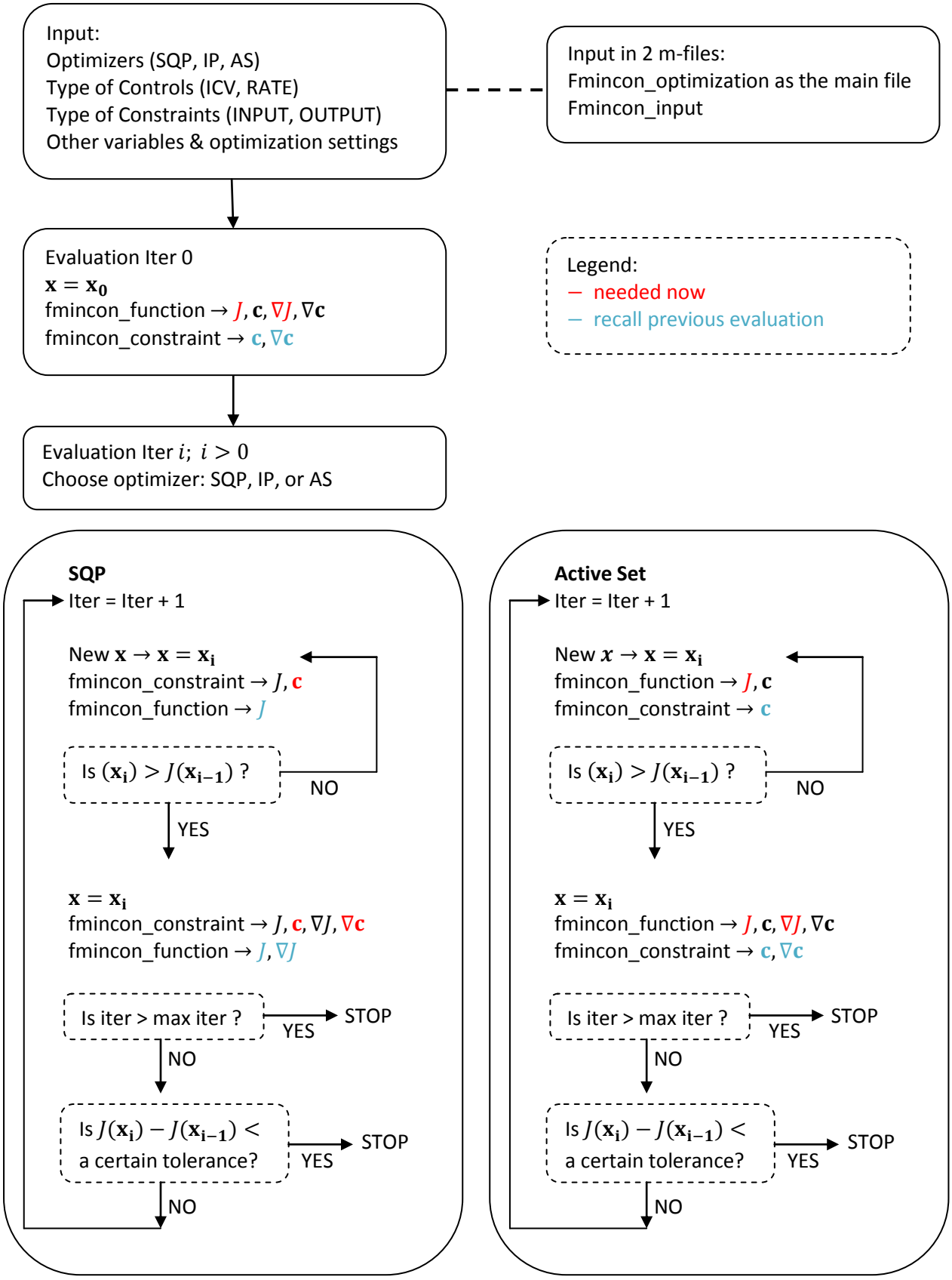
Szklarz, S., Rojas, M., and Kaleta, M. 2013. Efficient Solution of the Optimization Problem in Model-reduced Gradient-based History Matching. Paper A27 ECMOR XIII presented at European Conference on the Mathematics of Oil Recovery, 10-13 September 2012, Biarritz, France.

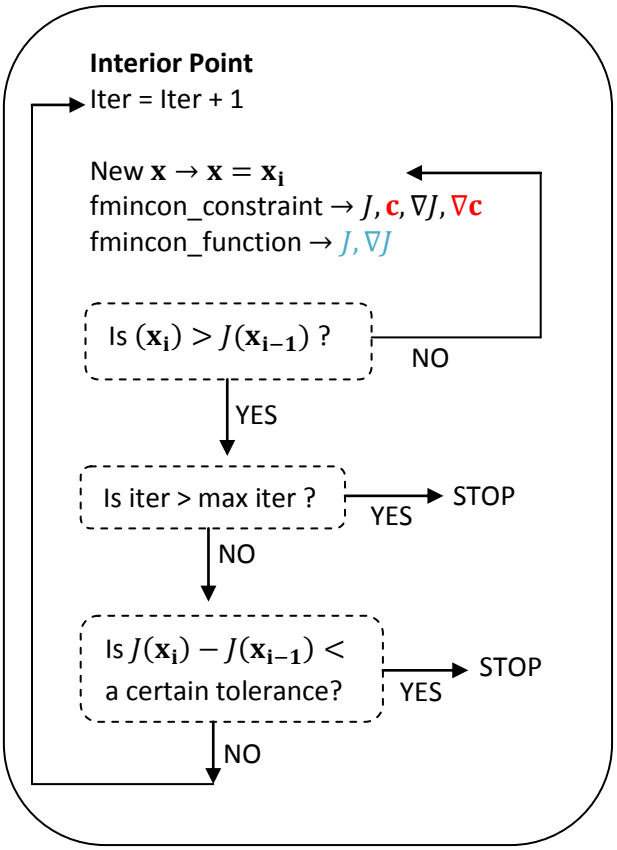
Van Essen, G.M., Zandvliet, M.J., Van den Hof, P.M.J., Bosgra, O.H. and Jansen, J.D., 2009: Robust water flooding optimization of multiple geological scenarios. SPE Journal 14 (1) 202-210. DOI: 10.2118/102913-PA.

Wen, T., Thiele, M.R., Ciaurri, D.E., Aziz, K., Ye, Y. 2013. Reservoir Management Using Two-stage Optimization with Streamline Simulation. Paper A33 ECMOR XIII presented at European Conference on the Mathematics of Oil Recovery, 10-13 September 2012, Biarritz, France.

APPENDIX

fmincon Matlab Code Flowchart




Interior Point
Iter = Iter + 1
New $\mathbf{x} \rightarrow \mathbf{x} = \mathbf{x_i}$
fmincon_constraint → J, $\mathbf{c}$, ∇J, $\nabla \mathbf{c}$
fmincon_function → J, ∇J
Is $(\mathbf{x_i}) > J(\mathbf{x_{i-1}})$?
NO
YES
Is iter > max iter ?
YES
STOP
NO
Is $J(\mathbf{x_i}) - J(\mathbf{x_{i-1}}) <$ a certain tolerance?
YES
STOP
NO