SPE 99959

Production Optimization With Adjoint Models Under Nonlinear Control-State Path Inequality Constraints

P. Sarma, Stanford U.; W.H. Chen, Chevron ETC; and L.J. Durlofsky and K. Aziz, Stanford U.

This paper was prepared for presentation at the 2006 SPE Intelligent Energy Conference and Exhibition held in Amsterdam, The Netherlands, 11–13 April 2006.

Abstract

The general petroleum production optimization problem falls under the category of optimal control problems with nonlinear control-state path inequality constraints (i.e. constraints that have to be satisfied at every time step), and it is acknowledged that such path constraints involving state variables are particularly difficult to handle. Currently, one category of methods implicitly incorporates the constraints into the forward and adjoint equations to tackle this issue. However, these are either impractical for the production optimization problem, or require complicated modifications to forward model equations (simulator). Thus, the usual approach is to formulate the above problem as a constrained nonlinear programming problem (NLP) where the constraints are calculated explicitly after the dynamic system is solved. The most popular of this category of methods (for optimal control problems) has been the penalty function method and its variants, which are, however, extremely inefficient. All other constrained NLP algorithms require the gradient of each constraint, which is impractical for an optimal control problem with path constraints, as one adjoint has to be solved for each constraint at each time step at every iteration.

We propose an approximate feasible direction NLP algorithm based on the objective function gradient and a combined gradient of the active constraints. This approximate feasible direction is then converted into a true feasible direction by projecting it onto the active constraints by solving the constraints during the forward model evaluation itself. The approach has various advantages. Firstly, only two adjoint evaluations are required at each iteration. Secondly, all iterates obtained are always feasible, as feasibility is maintained by the forward model itself, implying that any iterate can be considered a useful solution. Thirdly, large step sizes are possible during the line search, which can lead to significant reductions in forward and adjoint model evaluations and large reductions in the objective function. Through two examples, we demonstrate that the algorithm provides a practical and efficient strategy for production optimization with nonlinear path constraints.

Introduction

One of the primary goals of the reservoir modeling and management process is to enable decisions that maximize the production potential of the reservoir. Among the various existing approaches to accomplish this, real-time model-based reservoir management, also known as the "closed-loop" approach, has recently generated significant interest. This methodology entails model-based optimization of reservoir performance under geological uncertainty, while also incorporating dynamic information in real-time, which acts to reduce model uncertainty (see Figure 1). For such schemes to be practically applicable, a number of algorithmic advances are required. Some of our earlier papers [1,2], and also papers by other authors such as Brouwer et al. [3] have discussed efficient algorithms for such closed-loop production optimization.

This paper, however, is only focused on the optimization component of the closed-loop process, which is essentially a large-scale optimal control problem. A large variety of methods for solving discrete-time optimal control problems now exist in control theory literature, including dynamic programming, neighboring extremal methods, gradient-based nonlinear programming methods (NLP), etc. These are discussed in detail in Stengel [4] and Bryson and Ho [5]. Of these approaches, the NLP method combined with the *Maximum Principle* [5] (adjoint models) generates a class of NLP methods in which only the control variables are the decision variables and the state variables are obtained from the dynamic equations. These algorithms are generally considered more efficient compared to the other methods. Further, within this class of NLP methods, there are many existing techniques available for handling nonlinear control-state path inequality constraints [5,6,7,8]. However, as will be discussed later, these are either not practical for the production optimization problem or are difficult to implement with existing reservoir simulator codes.

In petroleum engineering literature, papers by various authors such as Asheim [9], Vironovsky [10], Brouwer and Jansen [11], etc. have discussed the application of adjoint models and gradient techniques for the production optimization problem in significant detail. However, an

important element that is missing from most of these papers is the treatment of nonlinear control-state path inequality constraints (for example, a maximum water injection rate constraint) effectively. Such constraints are always present in practical production optimization problems, and therefore their efficient treatment is essential for such algorithms to be useful. In one of our earlier papers [12], two methods to handle such constraints were discussed; however, they either do not satisfy the constraints exactly or are applicable only for small problems. In petroleum engineering literature, a paper by Zakirov et al. [13] discusses an approach to implement path constraints; however, there are certain theoretical issues with the approach, as discussed in a later section.

In this paper, we propose an approximate feasible direction optimization algorithm suitable for large-scale optimal control problems that is able to handle nonlinear inequality path constraints effectively while always maintaining feasibility. Other advantages are that only two adjoint simulations are required at each iteration and large step sizes are possible during the line search at each iteration, leading possibly to large reductions in the objective function. This method belongs to the class of NLP methods combined with the *Maximum Principle* (adjoint models) discussed above.

This paper proceeds with a brief description of the mathematical formulation of the problem and the application of adjoint models for efficient calculation of gradients of the objective function with respect to the controls. This is followed by a discussion of the existing methods for handling nonlinear path constraints for optimal control problems, with a critical comparison of their advantages and disadvantages. The next section discusses the traditional feasible direction optimization algorithm in some detail, as it is the basis of the proposed algorithm. This is followed by detailed discussions of the proposed approximate feasible direction and feasible line search algorithms. The validity and effectiveness of the proposed algorithm for handling nonlinear path inequality constraints is demonstrated through two examples, one with a maximum water injection constraint, and the other with a maximum liquid production constraint, both of which are nonlinear with respect to the controls (BHPs in this case).

Production Optimization with Adjoint Models

The production optimization problem under uncertainty requires finding a sequence of control vectors u^n for $n = 0,1,...,N-1$, where n is the control step index and N is the total number of control steps (here, for the purpose of clarity, time steps and control steps are considered equivalent), to maximize (or minimize) a performance measure $J\left(u^0,...,u^{N-1}\right)$. Our procedure for this optimization is discussed in detail in [1,12] so our description here will be brief. The problem definition is given by Equation (1).

Here, x^n refers to the dynamic states of the system, such as pressures, saturations, compositions etc. The cost function J consists of two terms. The first term ϕ is a function only of the dynamic states of the last control step (e.g., abandonment cost). The second term is a summation over all control steps and consists of the kernel L^n, which is known as the Lagrangian in control literature. Here it will involve the oil and water rates at each time step. Since L^n usually consists of well parameters or quantities that are functions of well parameters, it is written here in a fully implicit form.

$$\max_{u^n}\left[J\left(u^0,...,u^{N-1}\right) = \phi\left[x^N\right] + \sum_{n=0}^{N-1} L^n\left(x^{n+1},u^n\right)\right] \qquad \forall n \in (0,..,N-1)$$

subject to:

$$\begin{aligned} & g^n(x^{n+1},x^n,u^n) = 0 && \forall n \in (0,..,N-1) \\ & x^0 = x_0 && \text{(Initial Condition)} \\ & c^n(x^{n+1},u^n) \le 0 && \forall n \in (0,..,N-1) \\ & c^n(x^{n+1}) \le 0 && \forall n \in (0,..,N-1) \\ & LB \le u^n \le UB && \forall n \in (0,..,N-1) \end{aligned} \qquad (1)$$

The set of equations g^n together with the initial condition define the dynamic system. In the current application, g^n is the fully implicit reservoir simulation equations written for each grid block at each time step. The last three equations of Equation (1) define additional constraints for the controls – nonlinear inequality path constraints that are functions of both states and controls, nonlinear inequality path constraints that are functions of states only, and bounds on the controls. These equations constrain the controls directly, as opposed to the simulation equations that constrain only the dynamic states. Note that these are called path constraints because they have to be satisfied at every time step. Further, only inequality constraints are considered, because, almost all constraints in practical problems are inequality constraints. Examples of such constraints are maximum injection rate constraint, maximum water-cut constraint, maximum liquid production rate constraint, etc. Note that whether a constraint is linear or nonlinear also depends on the choice of control variables. For example, well rates are nonlinear functions of the BHPs of wells, and hence any rate constraint will be a nonlinear path constraint if BHPs are controlled; but may become a linear constraint if well rates are controlled directly. The control-state constraints and state only constraints are written separately because in general, state only constraints are more difficult to handle, and some existing algorithms treat them in different ways. In our proposed method, however, both of them will be treated with one unified approach.

In order to perform the optimization component of the closed loop (marked in blue in Figure 1) with gradient-based methods, an efficient approach to calculate the gradients of the cost function $J\left(u^0,...,u^{N-1}\right)$ with respect to the controls u^n is required. The most efficient method to calculate these gradients is through the use of the adjoint equations. The adjoint equations are obtained from the necessary conditions of optimality of the problem defined by Equation (1). The essence of the theory is that the cost function of Equation (1) along with all the constraints can be written equivalently in the form of an augmented cost function J_A:

$$J_A = \phi\left[x^N\right] + \sum_{n=0}^{N-1} L^n\left(x^{n+1}, u^n\right) + \sum_{n=0}^{N-1} \lambda^{T(n+1)} g^n\left(x^{n+1}, x^n, u^n\right) \quad (2)$$

For the moment, only the dynamic equations are considered. Treatment of the other constraints is discussed later. The vectors λ^n are known as Lagrange multipliers and one Lagrange multiplier is required for each constraint with which the cost function is augmented. Using *Calculus of Variations* [5], the first variation of the above equation must be zero for optimality, which gives the final form of the adjoint equations as follows:

$$\lambda^{Tn} = -\left[\frac{\partial L^{n-1}}{\partial x^n} + \lambda^{T(n+1)}\frac{\partial g^n}{\partial x^n}\right]\left[\frac{\partial g^{n-1}}{\partial x^n}\right]^{-1} \quad \forall n = 1,\ldots,N-1$$
$$\lambda^{TN} = -\left[\left.\frac{\partial \phi}{\partial x}\right|_{x=x^N} + \frac{\partial L^{N-1}}{\partial x^N}\right]\left[\frac{\partial g^{N-1}}{\partial x^N}\right]^{-1} \quad \text{(Final Condition)} \quad (3)$$

Because the λ^n depend on λ^{n+1}, the adjoint model is solved backwards in time, with the second equation above providing λ at the last time step (i.e., the initial condition for the backward integration). After solving for the Lagrange multipliers, the gradients of the cost function with respect to the controls can be calculated as:

$$\frac{dJ}{du^n} = \left[\frac{\partial L^n}{\partial u^n} + \lambda^{T(n+1)}\frac{\partial g^n}{\partial u^n}\right] \quad \forall n \in (0,\ldots,N-1) \quad (4)$$

These gradients can be used with any gradient-based algorithm to determine the new search direction and step length and thereby the new u^n. The process is repeated until the gradients are close to zero, implying that the optimum solution has been found. This is a very efficient procedure, as the time required to solve the adjoint model (and to calculate all of the required gradients) is about the same as that needed for the forward simulation (see [1,12] for full details). Note that the above derivation does not incorporate the nonlinear path constraints or the bounds on the controls. In the absence of such additional constraints, the above gradients can be used directly with any optimization algorithm. However, if additional constraints are present, their effect on the optimization has to be taken into account. The treatment of these constraints is discussed in the following sections.

Existing Methods for Nonlinear Path Constraints

It is acknowledged in control theory literature [5] that nonlinear control-state path inequality constraints involving state variables are the most difficult to incorporate effectively into optimal control algorithms. Even the *Maximum Principle* as given by Pontryagin does not apply directly to such problems [5]. As discussed earlier, among the various classes of algorithms available for optimal control problems, gradient-based NLP algorithms combined with adjoint models are generally considered the most efficient, and there are a number of existing NLP algorithms designed for path constrained optimal control problems. This class of NLP algorithms can be further divided into two categories: (1) algorithms that solve the path constraints implicitly together with the dynamic system or convert them to simple bounds constraints, implying that the NLP becomes an unconstrained NLP, thus constraint gradients are not required (2) algorithms that calculate the path constraints explicitly after the dynamic system has been solved, implying that the NLP becomes a constrained NLP, thus constrained NLP algorithms are required. The first four algorithms discussed next belong to the first category, and the last two belong to the second category.

One of the early methods given by Bryson et al. [5,14,15] incorporates the path constraints into the forward and adjoint equations in a manner similar to the dynamic system equations. In other words, the cost function is not only augmented with the dynamic system equations (see Equation (2)), but also with the path constraints, using an additional set of Lagrange multipliers. For example, if only control-state constraints are present, the augmented cost function is given by the following equation, where μ^n are the additional Lagrange multipliers.

$$J_A = \phi\left[x^N\right] + \sum_{n=0}^{N-1} L^n\left(x^{n+1}, u^n\right) + \sum_{n=0}^{N-1}\left\{\lambda^{T(n+1)} g^n\left(x^{n+1}, x^n, u^n\right) + \mu^{T(n+1)} c^n\left(x^{n+1}, u^n\right)\right\} \quad (5)$$

There is, however, an additional requirement on the Lagrange multipliers μ^n; see Bryson and Ho [5] for details. The numerical algorithm essentially involves joining together the constrained and unconstrained parts of the trajectories using the necessary conditions of optimality. This method is capable of finding the exact optimum quite efficiently and solves the path constraints implicitly together with the dynamic system, implying that solutions obtained at any iteration are always feasible. However, a major drawback of this technique is that the sequence of constrained and unconstrained parts of the trajectories at the optimum must be known apriori, and this information is not available for production optimization problems. Further, control-state and state only path constraints are not treated in the same manner, and state only constraints are generally quite difficult to implement with this approach [5].

Another method known as the *Generalized Gradient* method given by Mehra and Davis [6] shows that the difficulties associated with the method given by Bryson et al. [5,14,15] can be avoided by choosing different combinations of the control and state variables as the independent variables, instead of always choosing the control variables as independent variables. The different combinations of the control and state variables as independent variables are dictated by the constraints and could result in different combinations along different parts of the trajectory. The gradient of the cost function with respect to the independent variables, called the generalized gradient, is calculated by solving a set of equations similar to the Euler-Lagrange

equations [6]. Unfortunately, there are a few disadvantages of this method as well. The path constraints are only calculated explicitly after the dynamic system has been solved, resulting in possible infeasible solutions, in which case one has to start from a new guess. More importantly, since this method requires the ability to independently control the state variables, implementing this method is difficult in conjunction with existing simulator codes. For the same reason, standard NLP software cannot be easily used with this approach.

Yet another method known as the slack variable method (Jacobson and Lele [16]; Feehery [7]) essentially changes the original inequality constraint to an equality constraint by means of a slack variable *a(t)*:

$$c^n(x^{n+1},u^n)+\frac{1}{2}\left(a^n\right)^2=0 \qquad \forall n\in(0,..,N-1) \qquad (6)$$

The slack variable is squared so that any value of *a* is admissible. The principle of the method is to make one of the control variables appearing in the constraint a new state variable for the corresponding constraint. Thus, the constraint is appended as an equality constraint to the dynamic system, the control variable is solved as a state variable during the forward solve and the slack variable *a(t)* becomes the new control variable. Although the method is appealing and efficient, and like the first method, only generates feasible solutions, it again has some drawbacks, the main one being that, as the method changes the category of a control variable to a state variable, this means that any possible bounds on control variables cannot be satisfied. Unfortunately, almost all control variables for production optimization problems have either physical or economic bounds. Also, if more than one control variable is a candidate for conversion to a state variable, no suitable selection strategy exists. In practice, since a combination of control variables influences the state constraint, the method is not able to choose the correct one. Further, this method also requires significant modifications to existing simulation code.

In petroleum engineering literature, Zakirov et al. [13] have proposed an algorithm for implementing inequality path constraints. Their method is similar to that of Bryson et al. [5,14,15] in the sense that the active constraints are adjoined to the cost function by an additional set of Lagrange multipliers. The gradient of the objective function with respect to the independent controls is then calculated using the usual Euler-Lagrange equations (adjoint system). In order to calculate the step size, it is required that none of the constraints is violated by taking that step in the search direction. To do this, a problem of variations is solved. However, in order to do so, it is implied that the constraints that are active at a given iteration will also remain active in the next and succeeding iterations. There is no reason why this should be true, and such a scheme would result in the problem becoming overly constrained with succeeding iterations. It is also not clear how it is determined which constraints will be active at the optimum, and which controls will be kept independent.

The algorithms considered above solve the constraints implicitly together with the dynamic system. At the other extreme are algorithms that calculate the constraints explicitly after the dynamic system (forward model) has been solved, in order to determine whether the constraints were violated or not. The most popular among this class of algorithms for optimal control problems has been the penalty function approach and its variants [5, 17]. Penalty function methods transform the constrained optimization problem into alternative formulations such that the numerical solutions are sought by solving a sequence of unconstrained optimization problems [17]. For example, the production optimization as given by Equation (1) is converted to the following problem:

$$\begin{aligned}&\max_{u^n}\Phi_k\left(u^0,..,u^{N-1},r_k\right)\\&=J\left(u^0,..,u^{N-1}\right)+r_k\sum_{n=1}^{N}\Upsilon\left[c^n\left(x^{n+1},u^n\right)\right]\forall n\in(0,..,N-1)\\&\text{subject to:}\\&g^n(x^{n+1},x^n,u^n)=0 \qquad \forall n\in(0,..,N-1)\\&x^0=x_0 \qquad \text{(Initial Condition)}\\&LB\le u^n\le UB \qquad \forall n\in(0,..,N-1)\end{aligned} \qquad (7)$$

Here, $\Upsilon\left[c^n\left(x^{n+1},u^n\right)\right]$ is some function of the constraint $c^n\left(x^{n+1},u^n\right)$ and r_k is a positive constant known as the penalty parameter. With respect to $u^0,..,u^{N-1}$, the above optimization problem is an unconstrained optimization problem (except for the simple bounds which can be satisfied easily). If the unconstrained optimization of the function Φ_k is repeated for a sequence of values of r_k, the solution may be brought to converge to the solution of the original problem. The main reason for the popularity of the penalty function method within this class of methods (where constraints are calculated explicitly) is because constrained NLP becomes an unconstrained NLP, implying that only one gradient (that of the Φ_k) is required at each iteration; thus only one adjoint system has to be solved at each iteration. All other constrained NLP algorithms require the gradient of all active constraints at each iteration, and since for a path constrained optimal control problem, the number of active constraints and therefore the number of adjoint solves could be as large as the number of time steps of the forward problem (or more, if more than one path constraint is present), these algorithms are not practical for the production optimization problem. However, a key disadvantage of the penalty function method, which renders it impractical for the production optimization problem, is its inefficiency. Specifically, a large number of iterations for various values of r_k are typically required for significant improvement of the objective function. The problem is more severe when the initial guess is close to the constraint boundaries and one or more constraints are active at the optimum, and this is often the case with production optimization problems.

Due to the fact that the constrained NLP methods require the gradients of all active constraints, and thus become very expensive for path constrained optimal control problems, one

approach is to construct a single constraint from all the constraints, such that satisfying this single equivalent constraint ensures that all constraints will be satisfied. Such an approach is called constraint lumping [18]. With this approach, only two gradient calculations and therefore only two adjoint evaluations will be required at each iteration (one for the objective function and one for the equivalent constraint), which is a huge improvement compared to retaining all constraints. Various lumping schemes are available in the literature [18,8], with the following being commonly used for optimal control problems:

$$\sum_{n=0}^{N-1} \max\left[c^n(x^{n+1},u^n),0\right]=0$$
$$\sum_{n=0}^{N-1} \left\{\max\left[c^n(x^{n+1},u^n),0\right]\right\}^2=0 \qquad (8)$$

The second approach is an improvement over the first because the gradient of the first is discontinuous, although the second representation is also more nonlinear. Another approach is to create a smooth approximation to the *max* function, as given by Jennings et al. [8]:

$$\sum_{n=0}^{N-1} h\left[c^n(x^{n+1},u^n)\right]<\varepsilon; \quad \text{where}$$
$$h\left(c^n\right)=\begin{cases}0 & \text{if} \quad c^n\le -\varepsilon \\ \dfrac{\left(c^n+\varepsilon\right)^2}{4\varepsilon} & \text{if} \quad -\varepsilon\le c^n\le \varepsilon \\ c^n & \text{if} \quad c^n\ge \varepsilon\end{cases} \qquad (9)$$

The advantage of Equation (9) over the second equation of Equation (8) is the elimination of squaring, implying small deviations are penalized more heavily. In the algorithm proposed here, constraint lumping is applied in conjunction with the feasible direction algorithm. To our knowledge, this combination of algorithms has not been used previously for optimal control problems. Also, the lumping scheme used is a relatively new procedure [19], is more rigorous compared to Equation (9) and has not been used for optimal control problems before. The feasible direction algorithm and the particular constraint lumping scheme used are discussed in the next sections.

Feasible Direction Optimization Algorithm

The basic idea behind the method of feasible directions is to start from a feasible point (a point that satisfies all constraints) and move to a better point (with a lower objective function value) according to the iterative scheme [17]:

$$\mathbf{u_{i+1}}=\mathbf{u_i}+\beta S_i \qquad (10)$$

Note that $\mathbf{u}$ here represents the entire set of controls $u^0,..,u^{N-1}$, and $\mathbf{u_i}$ is the starting point for the i^{th} iteration, S_i is the search direction, β is the step length, and $\mathbf{u_{i+1}}$ is the final point obtained at the end of the iteration. The search direction S_i is found such that the following two properties are satisfied (1) a small move in the direction violates no constraint, and (2) the value of the objective function is reduced in that direction. The iterative process is repeated until no search direction can be found satisfying both properties. The final iterate represents a constrained local minimum of the problem. A direction satisfying both above-mentioned properties is called a usable feasible direction [17].

To exemplify, consider the optimization problem depicted by Figure 2. The figure shows the objective function contours (orange lines, dot-dash) and three constraints (green lines, solid), which are functions of two control variables only. No state variables are present for simplicity. The blue dot labeled "optimum" depicts the constrained minimum of the problem, constrained by c_3. Thus only constraint c_3 is active at the optimum. The initial guess or starting point is at the intersection of constraints c_1 and c_2, meaning that c_1 and c_2 are active at the initial guess. The blue arrow (dashed) is the negative of the objective function gradient, and the purple arrows (dashed) are the negatives of the active constraint gradients. The thick orange lines represent the cone of feasibility at the initial guess, that is, any search direction within this cone will satisfy property 1. The pink arrow (solid) is a usable feasible direction, that is, a search direction that satisfies both properties 1 and 2. Such a direction at a point $\mathbf{u_i}$ can be determined mathematically by a direction S_i that satisfies the following equations [17]:

$$\left.\frac{d}{d\beta}J\left(\mathbf{u_i}+\beta S_i\right)\right|_{\beta=0}=S_i^T\nabla J\left(\mathbf{u_i}\right)\le 0$$
$$\left.\frac{d}{d\beta}c^n\left(\mathbf{u_i}+\beta S_i\right)\right|_{\beta=0}=S_i^T\nabla c^n\left(\mathbf{u_i}\right)\le 0 \qquad (11)$$
$$\forall n=0,...,N-1$$

Here, $\nabla J\left(\mathbf{u_i}\right)$ and $\nabla c^n\left(\mathbf{u_i}\right)$ are the gradients of the objective function and the active constraints at point $\mathbf{u_i}$. It is possible to reduce the objective function J at least by a small amount by taking a step length $\beta>0$ along such a direction.

There are many different ways to determine a usable feasible direction, giving rise to different feasible direction algorithms. In the current work, the well-known Zoutendijk's method of feasible directions is used [17], and will therefore be discussed here. In this method, the usable feasible direction at the current iterate is taken as the negative of the objective function gradient if the iterate lies in the interior of the feasible region. However, if it lies on the constraint boundary (or close to the boundary within some pre-set tolerance), a usable feasible direction that satisfies Equation (11), is found by solving a linear programming problem [17]:

$$\begin{aligned} &\min_{S_i} \quad -\gamma \\ &\text{subject to:} \\ &S_i^T \nabla c^n(\mathbf{u}_i) + \theta_n \gamma \le 0 \qquad n = 0,\ldots,p-1 \\ &S_i^T \nabla J(\mathbf{u}_i) + \gamma \le 0 \\ &-1 \le s_{i,k} \le 1 \end{aligned} \tag{12}$$

Here, $s_{i,k}$ is the k^{th} component of S_i (recall that i designates iteration), and the first p constraints are assumed to be active (or almost active) at point $\mathbf{u_i}$ (the constraints can always be renumbered to satisfy this requirement). Here γ is taken as an additional design variable. Any value of $\gamma > 0$ would provide a usable feasible direction S_i. The maximum value of γ gives the best direction S_i that makes the value of $S_i^T \nabla J(\mathbf{u_i})$ maximally negative and the values of $S_i^T \nabla c^n(\mathbf{u_i})$ as negative as possible simultaneously. In other words, the maximum value of γ makes the direction S_i steer away from the active nonlinear constraints. Different values of θ_n for different constraints allow us to give more importance to certain constraint boundaries as compared to others. For more details, refer to Rao [17].

The feasible direction algorithm is useful when the initial guess point is at (or close to) constraint boundaries and the steepest descent direction (negative objective function gradient direction) is pointing away from the feasible region. This is usually the case with production optimization problems. In such a case, first order algorithms (like the steepest descent algorithm) or even quasi-Newton algorithms would provide search directions moving along which would violate one or more constraints, thereby providing infeasible iterates.

Approximate Feasible Direction Algorithm

Zoutendijk's method of feasible directions [17], like all other constrained NLP algorithms, requires the gradients of all active constraints. As seen in Equation (12), these gradients are required to calculate the feasible direction. However, as mentioned earlier, this is not practical for an optimal control problem with path constraints due to the excessive number of adjoint evaluations required to calculate these gradients. One approach to alleviate this problem is to apply constraint lumping to create one equivalent constraint from all active constraints, with the benefit that only two adjoint solutions will be required at each iteration, one for the objective function and one for the equivalent constraint. The particular lumping scheme used in this work is given by Liu et al. [19] and is a relatively new approach. It is essentially a smooth differentiable approximation of the *max* function, but is more rigorous compared to Equation (9). In their work, the lumping scheme is used within a penalty function method, whereas in this work we apply it in conjunction with the feasible direction method, which leads to a different interpretation.

Consider the discontinuous unit-step function and its approximation, the sigmoid function, given by the following equations:

$$\begin{aligned} &\sigma(y) = \begin{cases} 1 & \text{if } y > 0 \\ 0 & \text{if } y \le 0 \end{cases} \\ &s(y,\alpha) = \left\{1 + \exp(-\alpha y)\right\}^{-1} \; \forall \alpha > 0 \end{aligned} \tag{13}$$

The *max* function is an integral of the unit-step function and is given by the following equation:

$$\max\{x,0\} = \int_{-\infty}^{x} \sigma(y)\,dy \tag{14}$$

Substituting $s(y,\alpha)$ for $\sigma(y)$ in the expression above, it can be shown that the following equation approximates the *max* function [19]:

$$p(x,\alpha) = \int_{-\infty}^{x} s(y,\alpha)\,dy = x + \frac{1}{\alpha}\log\left\{1 + \exp(-\alpha x)\right\} \tag{15}$$

The function $p(x,\alpha)$ has infinitely many continuous derivatives. Some other properties of the function $p(x,\alpha)$ relevant to its application with the feasible direction algorithm are as follows:

$$\begin{aligned} &p(x,\alpha) > \max\{x,0\} \qquad \forall x \in R \\ &\lim_{|x|\to\infty} \left\{p(x,\alpha) - \max(x,0)\right\} = 0 \;\; \forall \alpha > 0 \\ &\lim_{\alpha\to\infty} \left\{p(x,\alpha) - \max(x,0)\right\} = 0 \;\; \forall x \in R \end{aligned} \tag{16}$$

Due to the above properties, the function $p(x,\alpha)$ can be used as an approximation of the *max* function for constraint lumping. This circumvents the main disadvantage of the *max* function, which is its non-differentiability (this makes it difficult to implement the *max* function with gradient-based algorithms). However, due to the infinite differentiability property mentioned above, $p\left\{c^n(x^{n+1},u^n),\alpha\right\}$ will be as many times differentiable as $c^n(x^{n+1},u^n)$. Figure 3 shows the *max* function and its approximation with $p(x,\alpha)$ for various values of α. Therefore, the equivalent constraint replacing the *max* constraint lumping scheme for the path constraints of Equation (1) is given as:

$$C = \sum_{n=0}^{N-1}\left[c^n + \frac{1}{\alpha}\log\left\{1 + \exp\left(-\alpha c^n\right)\right\}\right] \le \frac{\log 2}{\alpha} \quad \forall \alpha > 0 \tag{17}$$

The $\log 2/\alpha$ term appears in the above equation because $p(c^n = 0,\alpha) = \log 2/\alpha$ and $p(c^n,\alpha)$ increases monotonically with c^n. With this definition, the optimal control problem equivalent of Equation (1) is given by Equation (18) below. The nonlinear path constraint has been replaced by the single integral constraint *C*. The bounds on the controls are still present, but they can be easily satisfied using standard techniques like gradient projection.

$$\max_{u^n}\left[\, J = \phi\left[x^N\right] + \sum_{n=0}^{N-1} L^n\left(x^{n+1}, u^n\right)\right] \forall n \in (0,..,N-1)$$
subject to:
$$\begin{aligned} & g^n(x^{n+1}, x^n, u^n) = 0 && \forall n \in (0,..,N-1) \\ & x^0 = x_0 && \text{(Initial Condition)} \\ & C = \sum_{n=0}^{N-1}\left[c^n + \frac{1}{\alpha}\log\left\{1+\exp\left(-\alpha c^n\right)\right\}\right] \le \frac{\log 2}{\alpha} \\ & LB \le u^n \le UB && \forall n \in (0,..,N-1) \end{aligned} \quad (18)$$

From the perspective of the feasible gradient algorithm, since the function $p\left\{c^n\left(x^{n+1}, u^n\right), \alpha\right\}$ is an approximation of $\max\left\{c^n\left(x^{n+1}, u^n\right), 0\right\}$, it is clear that the equivalent constraint C is essentially a sum of the active constraints, for large enough α. Therefore, the gradient of the equivalent constraint C is the sum of the gradients of the active constraints. This is demonstrated in Figure 4, where the dashed brown arrow is the sum of the gradients of the active constraints c_1 and c_2, and these gradients themselves are depicted by the dashed purple arrows. Therefore, the implication of using this particular constraint lumping as opposed to directly solving the original problem is that, instead of obtaining the gradients of all the active constraints individually, only a single gradient direction is obtained, which is the sum of the gradients of all active constraints. With only this single direction, the apparent feasibility cone is given by the thick gray line. This apparent feasibility cone will always be equal to or wider than the true feasibility cone (thick orange line). Again, a linear program can be solved to determine the feasible direction:

$$\min_{s_i} \quad -\gamma$$
subject to:
$$\begin{aligned} & S_i^T \nabla C(\mathbf{u}_i) + \theta\gamma \le 0 \\ & S_i^T \nabla J(\mathbf{u}_i) + \gamma \le 0 \\ & -1 \le s_{i,k} \le 1 \end{aligned} \quad (19)$$

Note that, because the apparent feasibility cone may be wider than the true feasibility cone, the feasible direction obtained may not be actually feasible. For example, using Equation (19), a direction such as the solid pink arrow (Figure 4) may be obtained as the feasible direction. Although this direction is within the apparent feasibility cone, it is outside the true feasibility cone, and is therefore not truly a feasible direction. Moving in this direction even infinitesimally would result in the violation of constraint c_2 (but not c_1). This direction is therefore called an approximate feasible direction. However, this direction will usually be better than just the steepest descent direction (negative objective function gradient). For example, as seen in Figure 4 moving in the steepest descent direction violates both active constraints.

In order to solve the above problem of approximate feasibility, a feasible line search algorithm is employed. The key idea behind the feasible line search algorithm is to implement the path constraints within the forward model and modify the search direction within the forward model if the path constraints are violated. The approximate feasible direction is thus projected onto the infeasible active constraints during line-search by solving the constraints during the forward simulation. Note that projecting this direction onto the infeasible active constraints during line-search is equivalent to performing a "curved" line search along the infeasible active constraints, as seen in Figure 5. Gradient information from previous iterations and our knowledge of the dynamic system can be used to determine which controls need to be modified to satisfy an infeasible constraint. For example, if a path constraint such as a maximum injection rate constraint is violated at a given time-step of the forward simulation, the controls associated with the injectors at that time-step will have the maximum influence on the constraint, and should therefore be modified to satisfy the constraint. Note that there could be many possible choices of controls or combinations thereof that can be modified to satisfy a constraint, and it is not clear at the moment if a particular "best" strategy exists that could be employed to choose the right controls.

In the current work, a maximum total water injection constraint and a maximum total liquid production constraint have been implemented. For the maximum total water injection rate constraint, if the constraint is violated at a given time-step, controls associated with all the injectors (BHPs) at that time-step are modified to satisfy the constraint. Similarly, for the maximum total liquid production constraint, the producer BHPs are modified to satisfy the constraint. In the current implementation, a simple, easy to implement iterative approach is used to determine the modified controls if the path constraints are violated at a given time-step. The approach assumes that a linear relationship exits between the injection rate of an injector (or liquid production rate of producer) and the pressure difference between its BHP and well block pressure, as depicted by the following equation:

$$p_w^{tgt} = p_b^{n+1} - \omega\frac{q^{tgt}}{q^{cur}}\left(p_b^{n+1} - p_w^{cur}\right) \quad (20)$$

To clarify, after solving the forward model at a given time-step with the values of the controls as provided by the optimization algorithm (approximate feasible direction algorithm in this case), if a constraint is violated (i.e., $q^{tgt} < q^{cur}$), new values of the controls to be modified (p_w^{tgt}) are obtained with the above equation using the current values of the controls (p_w^{cur}), and the forward model is solved again at the same time-step with the new control values. The process is repeated until the constraint is satisfied. ω is a relaxation factor used to accelerate convergence. This approach is certainly not the most efficient, and the best approach would be to solve the violated constraints together with the dynamic system.

The main benefits of the feasible line search algorithm are that all iterates obtained are always feasible, implying that any iterate can be considered a useful solution, and large step sizes

are possible during the feasible line search, leading to significant reductions in forward model evaluations and possibly also the objective function.

In order to account for the bounds on the controls, a projected gradient algorithm (with the approximate feasible direction as the search direction) is used. Refer to Kelley [20] for more details about gradient projection. The simulator used in this work is the General Purpose Research Simulator (GPRS) developed at Stanford [21], and the adjoint models are built directly from the simulator code, as described in [12].

Example 1 – Horizontal Smart Wells

The first case is a simple example adapted from Brouwer and Jansen [11] that effectively demonstrates the applicability of the proposed algorithm to smart well control with nonlinear path constraints. The schematic of the reservoir and well configuration is shown in Figure 6. The model consists of one horizontal "smart" water injector and one horizontal "smart" producer, each having 45 controllable segments. The reservoir covers an area of 450×450 m^2 and has a thickness of 10 m and is modeled by a 45×45×1 horizontal 2D grid. The fluid system is an essentially incompressible two-phase unit mobility oil-water system, with zero connate water saturation and zero residual oil saturation. The left figure of Figure 7 shows the heterogeneous permeability field with two high permeability streaks running from the injector (left) to the producer (right). The contrast in permeability between the high permeability streaks and the rest of the reservoir is around a factor of 20-40.

For purposes of optimization, the injector and producer segments are placed under BHP control. The objective of the optimization process is to maximize Net Present Value (NPV) of the reservoir (see [12] for definition). The NPV discounting factor is set to zero, meaning that the effect of discounting is neglected. Thus, maximizing NPV is essentially maximizing cumulative oil production and minimizing cumulative water production. To allow comparison with Brouwer and Jansen [11] and our earlier work [12], the oil price is conservatively set at \$80/m^3, water injection costs at \$0/m^3, and water production costs at \$20/m^3 [11]. There is a maximum total water injection constraint of 2710 STBD, which is a nonlinear path constraint in this case, because the controls are the BHPs of segments. Further, there are bounds on the BHPs of the segments, which could for example correspond to bubble point pressures or fracture pressures. The model is produced until exactly one pore volume of water is injected, which corresponds to around 950 days of injection. This time period is divided into five control steps of 190 days each. Thus the total number of controls is equal to (45 + 45)×5 = 450. In our earlier paper [12], the same example was studied with the injector segments under rate control, in which case the maximum injection rate constraint was a linear constraint. Thus, comparing against our earlier results, this case demonstrates the validity and effectiveness of the proposed approach to handle nonlinear path constraints.

The base case is a constant injection rate, constant producer BHP operation strategy. The injection rate is kept at the maximum of 2710 STBD and is distributed among the 45 injection segments according to their *kh*, which corresponds to an uncontrolled case. The producer BHPs are set in such a way that a balanced injection-production is obtained.

Starting from an initial oil saturation of 100% throughout the reservoir, Figure 7 (right) shows the final oil saturations for the uncontrolled case obtained after 950 days of production. Figure 8 shows the final oil saturations after 950 days for the optimized cases, the left image with the injector segments under rate control (from [12]), and the right image with the injector segments under BHP control (performed with the proposed algorithm). As discussed above, rate controlled injectors correspond to the injection rate constraint being a linear path constraint, and for BHP controlled injectors, the rate constraint becomes nonlinear. It is clear that the optimization leads to a large improvement in the sweep efficiency for both cases, and the proposed algorithm with BHP controlled injectors performs almost as well as the original algorithm [12] with linear constraints only, validating the effectiveness of the approach. The optimization leads to an increase in NPV of approximately 100%. Figure 9 shows that there is a substantial increase in cumulative oil production (70%), attributed to the better sweep, and a slight decrease in water production (6%) after the optimization process. The optimization process required 4 iterations and the total number of simulations (including adjoint simulations) required for the optimization was around 15.

Figure 10 shows the injection rates for the reference case and the optimized case (with BHP controlled injectors). It is clear that the constraint (2710 STBD) is satisfied to within 1% tolerance after optimization. Note that even after the optimization, the water injection rate remains near the maximum for most of the time, thus implying that the optimization essentially results in redistribution of the injected water among the injection segments.

The reasons behind the better sweep in the optimized case can be easily explained by analyzing the optimized trajectories of the controls, i.e., BHPs of the injectors and producers as seen in Figure 11. The *y*-axis of Figure 11 corresponds to 45 segments (injectors on the left and producers on the right) and the *x*-axis corresponds to the 5 control steps. The color scale corresponds to BHPs of the segments. It is obvious that the injector segments completed in or near the high permeability streaks are shut-in most of the time, as they would otherwise force water to move very quickly toward the producers, resulting in early breakthrough and thus poor sweep. For the producers, we again observe that the segments completed in or near the high permeability streaks are shut most of the time, in order to force the injected water into the low permeability regions to improve sweep.

Example 2 – Arab-D Formation, Ghawar Reservoir

The second example studied is a more realistic example adapted from Yeten [22] and Sengul et al. [23]. The simulation model is a conceptual representation of a small portion of the Arab-D formation of the Ghawar reservoir (see Figure 12 left). The 3D simulation model, populated with the initial oil distribution, is shown in Figure 13 (left). The red blocks indicate oil and blue blocks indicate water, while the blocks in between indicate the transition zone. The reservoir model consists of 25x33x10 cells with grid sizes in the *x* and *y* directions of 200 feet. The thickness of each layer varies, and

properties are provided in [22]. The movable oil originally in place is around 18 MMSTB. The model has aquifer support along the east flank. The other boundaries were modeled as no-flow.

This field is a naturally fractured carbonate reservoir. Fractures act as the fastest fluid flow paths within the reservoir. The matrix also has reasonable permeability and contributes significantly to fluid flow. Two distinct fracture distributions are identified within the field. Here they will be referred to as fractures and stratiform "Super - K" layers. The fractures are modeled as vertical high permeability zones, oriented along the east-west plane, cutting all layers from top to bottom. The stratiform Super - K layers are modeled as thin layers with high permeability. Figure 12 (right) shows how the fractures and the stratiform Super - K layers are oriented. The refined grids in the *y* direction in Figure 13 (left) represent the vertical fractures, and the thin layers (5 and 9) represent the Super - K layers. Additional properties are provided in [22].

Because the field is operated above the bubble point pressure, the simulation model only includes oil and water phases. A tri-lateral production well is completed in the second layer of the simulation model (Figure 13 right). All laterals as well as the main-bore are horizontal. The heel of the main-bore is highlighted with a full white circle on this plot. The branch closest to the heel of the well will be referred to as Branch 1, the one just below it will be referred to as Branch 2, and the last one will be referred to as Branch 3, as shown in Figure 13 (right). Note that Branch 2 intersects a fracture and Branch 1 is very close to a fracture. Branches 1 and 2 are about 2000 ft long and Branch 3 is about 3000 ft long. The branches are spaced approximately 1400 ft apart from each other, and all have open-hole completions. The laterals are fully perforated (no partial perforation) and the main-bore is not perforated.

The simulation model was run for 1800 days (approximately 5 years). Production is subject to a maximum liquid production constraint of 6000 STBD, and the controls are the BHPs of each lateral, thereby making the liquid production constraint a non-linear path constraint. The minimum allowable BHP at the heel of the well was set at 2600 psi (equivalent approximately to a WHP of 250 psi, as in [22]). The objective is to maximize the cumulative oil production of the reservoir. The base case is an uncontrolled case producing at the maximum liquid rate (6000 STBD). Uncontrolled implies that the BHPs of the laterals are not controlled directly, and they adjust themselves according to the production rate and the BHP at the heel of the well. Because the GPRS version used in this work did not have down-hole choke models implemented, the tri-lateral well was actually modeled as three separate horizontal wells. To maintain approximate consistency with Yeten's model [22], for the uncontrolled case the BHPs at the junctions of the laterals and the main-bore were specified based on the ECLIPSE [24] simulation results generated in [22]. This entailed specification of time-varying BHP for each lateral. The resulting oil rates and water cuts generated by GPRS are close to the original ECLIPSE results of Yeten [22], indicating that our base case is consistent with the earlier model.

Figure 14 shows the oil production profiles for individual branches. As explained in [22], the resulting production is unbalanced. In the uncontrolled (reference) case, Branch 1, which is closest to the heel of the well, produces more oil than the other two branches. Branch 2 produces significantly more than Branch 3, especially for the early times before the water breaks through, due to its intersection with a fracture. Figure 15 presents the water cut for each branch. The water cut of Branch 3 (reference) is much less compared to the others, likely due to the proximity of Branches 1 and 2 to the fractures and Super – K layers, which act as fast conduits to the aquifer. This results in unbalanced production and unbalanced sweep (Figure 16 left) which are detrimental to overall recovery.

Figure 14 also shows the oil production profiles after optimization with the proposed algorithm. The 1800 days of production was divided into 18 controls steps of 100 days each, resulting in a total of 18x3 = 54 controls. As would be expected, the algorithm allocates more production to Branch 3 than the other branches. It can also be seen that Branch 2 has been allocated the least amount of production due to its direct connection to a fracture. Figure 15 also shows the water cut profiles of the branches after optimization. The water breaks through in Branch 3 earlier. This water comes from the matrix and its water cut does not increase as rapidly as in the other branches. Therefore the breakthrough in Branch 3 does not affect the overall performance as much as the breakthrough in other branches. Figure 16 (right) also shows that the final sweep pattern obtained after optimization is more balanced, resulting in more oil being produced.

Figure 17 shows the field water cuts for the uncontrolled and optimized cases. It is clear that because there is a maximum liquid production constraint (which is essentially active in the uncontrolled case), the only way to increase cumulative oil production is to reduce the field water cut. It is observed from Figure 17 that the field water cut is higher for the optimized case for about 300 days, after which it reduces below the base case. The final water cut after 1800 days is reduced from 0.66 to 0.54, resulting in an increase in cumulative oil production of about 16% over the base case, as seen in Figure 18. The liquid production rates for the uncontrolled and the optimized cases are plotted in Figure 19. The optimization clearly satisfies the maximum constraint of 6000 STBD at all times, again validating the effectiveness of the proposed algorithm. The total number of iterations of the optimization algorithm was 11, and the total number of simulations required including adjoint simulations was 68. Yeten [22] used a conjugate gradient algorithm with numerical gradients for the optimization, with ECLIPSE as the simulator, and he also found a similar improvement in cumulative oil production. However, the number of simulations required with his approach will in general be much more due to the use of numerical gradients.

Conclusions

In this paper, an approximate feasible direction algorithm combined with a feasible line search was proposed to handle production optimization problems with nonlinear path inequality constraints, with the following major benefits:

1. Due to constraint lumping, only two adjoint evaluations are required at each iteration of the optimization algorithm, which is one of the reasons behind the efficiency of the algorithm.

2. All iterates obtained are always feasible, implying that the optimization may be stopped at any iteration and the final iterate can be considered a useful solution.

3. Large step sizes are possible during the modified line search, leading to significant reductions in forward model evaluations during the line search, and may also result in significant reduction of the objective function.

4. Problems associated with starting close to the boundary of the feasible region, which may limit the effectiveness of penalty function methods, are avoided. Such starting points often occur in production optimization problems.

The effectiveness and applicability of the algorithm was demonstrated through two examples: the first was a dynamic waterflood optimization problem with a maximum injection rate constraint, and the second was a tri-lateral well optimization problem with aquifer support and a maximum liquid production rate constraint. Both optimizations resulted in significant improvement in the objective functions (NPV and cumulative oil production), implying that model-based optimization has considerable potential for practical reservoir management.

Acknowledgements

We thank the industrial affiliates of the Stanford University Advanced Wells Research Consortium (SUPRI-HW), Saudi ARAMCO and Chevron for financial support of this work. We also thank Burak Yeten (Chevron ETC) for providing the simulation model for the Ghawar case used in this study.

Nomenclature

a	Slack variable
c	Nonlinear constraints
C	Equivalent constraint
g	Dynamic system equations
J	Cost function
J_A	Augmented cost function of original cost function
L	Lagrangian
LB	Lower bounds on controls
N	Number of control steps
p	Number of active constraints
p_b	Well block pressue
p_w	Well BHP
q	Fluid flow rate
r_k	Penalty parameter at k^{th} iteration
S	Usable feasible direction
t	Time
u	Control vector
$\mathbf{u}$	Comple set of $u^0,..,u^{N-1}$
UB	Upper bounds on controls
x	Dynamic states
α	Tolerance for *max* function approximation
β	Line search step length
Υ	Function of constraints
γ	Parameter to maximize
ε	Tolerance for max function approximation
ϕ	Part of cost function
Φ_k	Penalty function at k^{th} iteration
λ	Lagrange multipliers
μ	Lagrange multipliers
θ	Weight on constraints
ω	Relaxation factor

Subscripts

i	Iteration index
k	Vector component index

Superscripts

n	Time level n
T	Transpose of vector

Acronyms

BHP	Well bottom hole pressure
FLPR	Field liquid production rate
FOPT	Field oil production total
FWCT	Field water cut
FWPT	Field water production total
WHP	Wel head pressure
WOPR	Well oil production rate
WWCT	Well water cut

References

1. Sarma, P., Durlofsky, L.J., Aziz, K. and Chen, W.H., Efficient Real-time Reservoir Management Using Adjoint-based Optimal Control and Model Updating, *Computational Geosciences*, Vol. 9, No. 4, 2005
2. Sarma, P., Durlofsky L.J., and Aziz, K., Efficient Closed-loop Production Optimization under Uncertainty, SPE paper 94241 presented at the SPE Europec/EAGE Annual Conference, Madrid, Spain, 2005
3. Brouwer, D.R., Naevdal, G., Jansen, J.D., Vefring, E.H. and van Kruijsdiik, C.P.J.W., Improved Reservoir Management Through Optimal Control and Continuous Model Updating, SPE paper 90149 presented at the SPE Annual Technical Conference and Exhibition, Houston, TX, 2004
4. Stengel, R.F., *Optimal Control and Estimation* (Dover Books on Advanced Mathematics, New York, 1985)
5. Bryson, A.E. and Ho, Y.C., *Applied Optimal Control; Optimization, Estimation, & Control* (Taylor & Francis, New York, 1975)
6. Mehra, R.K. and Davis, R.E., A Generalized Gradient Method for Optimal Control Problems with Inequality Constraints and Singular Arcs, *IEEE Transactions on Automatic Control*, Vol. AC-17, No. 1, 1972
7. Feehery, W.F., *Dynamic Optimization with Path Constraints* (PhD Thesis, MIT, 1998)
8. Fisher, M.E. and Jennings, L.S., Discrete Time Optimal Control Problems with General Constraints, *ACM Transactions on Mathematical Software*, Vol. 18, No. 4, 1992
9. Asheim, H., Maximization of Water Sweep Efficiency by Controlling Production and Injection Rates, paper number SPE 18365 presented at the SPE European Petroleum Conference, London, 1988
10. Vironovsky, G.A., Waterflooding Strategy Design Using Optimal Control Theory, presented at the 6th European IOR Symposium, Stavanger, 1991
11. Brouwer, D.R. and Jansen, J.D., Dynamic Optimization of Water Flooding with Smart Wells using Optimal Control

Theory, paper number SPE 78278 presented at the SPE 13th European Petroleum Conference, Aberdeen, 2002
12. Sarma, P., Aziz, K. and Durlofsky, L.J., Implementation of Adjoint Solution for Optimal Control of Smart Wells, SPE paper 92864 presented at the SPE Reservoir Simulation Symposium, Houston, TX, 2005
13. Zakirov, I.S., Aanonsen, S.I., Zakirov, E.S. and Palatnik, B.M., Optimization of Reservoir Performance by Automatic Allocation of Well Rates, presented at the 5th European Conference on the Mathematics of Oil Recovery, Leoben, 1996
14. Bryson, A.E., Denham, W.F. and Dreyfus, S.E., Optimal Programming Problems with Inequality Constraints I: Necessary Conditions for Extremal Solutions, *AIAA Journal*, Vol. 1, No. 11, 1963
15. Denham, W.F. and Bryson, A.E., Optimal Programming Problems with Inequality Constraints II: Solution by Steepest Descent, *AIAA Journal*, Vol. 2, No. 11, 1964
16. Jacobson, D. and Lele, M., A Transformation Technique for Optimal Control Problems with a State Variable Inequality Constraint, *IEEE Transactions on Automtic Control*, Vol. 5, 457-464, 1969
17. Rao, S.S., *Optimization: Theory and Applications* (John Wiley and Sons Ltd, New York, 1978)
18. Agkun, M., Haftka, R.T. and Wu, K.C., Sensitivity of Lumped Constraints using the Adjoint Method, presented at the 40th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, St. Louis, MO, 1999
19. Liu, S.Q., Shi, J., Dong, J. and Wang, S., A Modified Penalty Function Method for Inequality Constraints Minimization, from http://www.mmm.muroran-it.ac.jp/~shi/
20. Kelley, C.T., *Iterative Methods for Optimization* (Society of Industrial and Applied Mathematics, Philadelphia, 1999)
21. Cao, H., Development of Techniques for General Purpose Simulators (PhD Thesis, Stanford University, 2002)
22. Yeten, B., Optimum Deployment of Nonconventional Wells (PhD Thesis, Stanford University, 2003)
23. Sengul, M., Yeten, B. and Kuchuk, F., The Completion Challenge – Modeling Potential Well Solutions, *Middle East and Asia Reservoir Review*, No. 5, 2004
24. Schlumberger, *ECLIPSE 100 Technical Description 2001A* (Schlumberger, 2001)
25. Jansen, J.D., Brouwer, D.R., Naevdal, G. and van Kruijsdiik, C.P.J.W., Closed-loop Reservoir Management, *First Break*, Vol. 23, 43-48, 2005

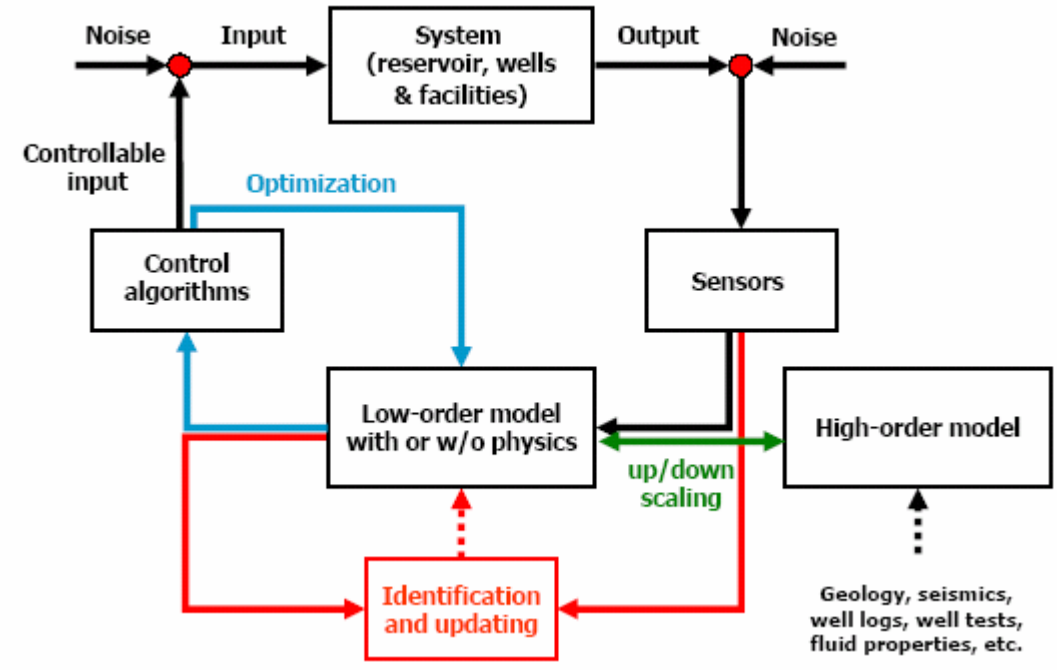

Figure 1 Schematic of the Closed Loop Optimal Control approach (from Jansen et al. [25]).

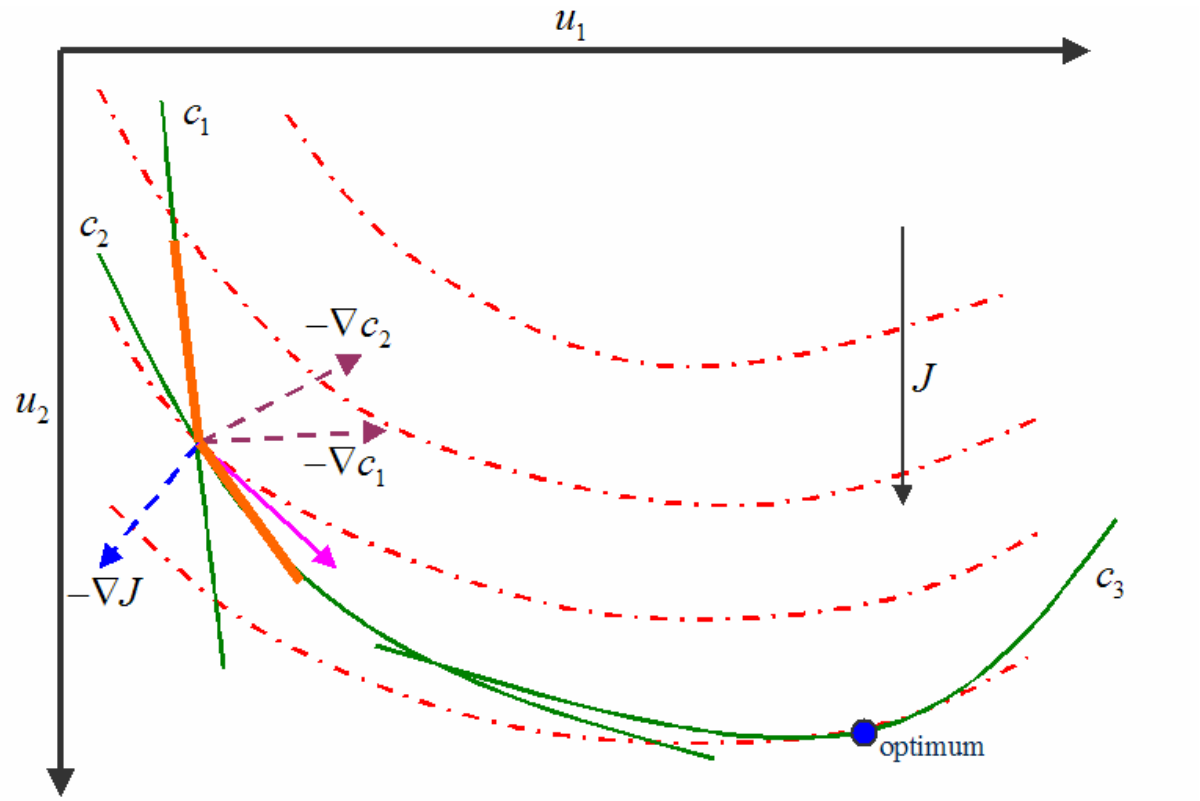

Figure 2 Schematic of a simple optimization problem with constraints.

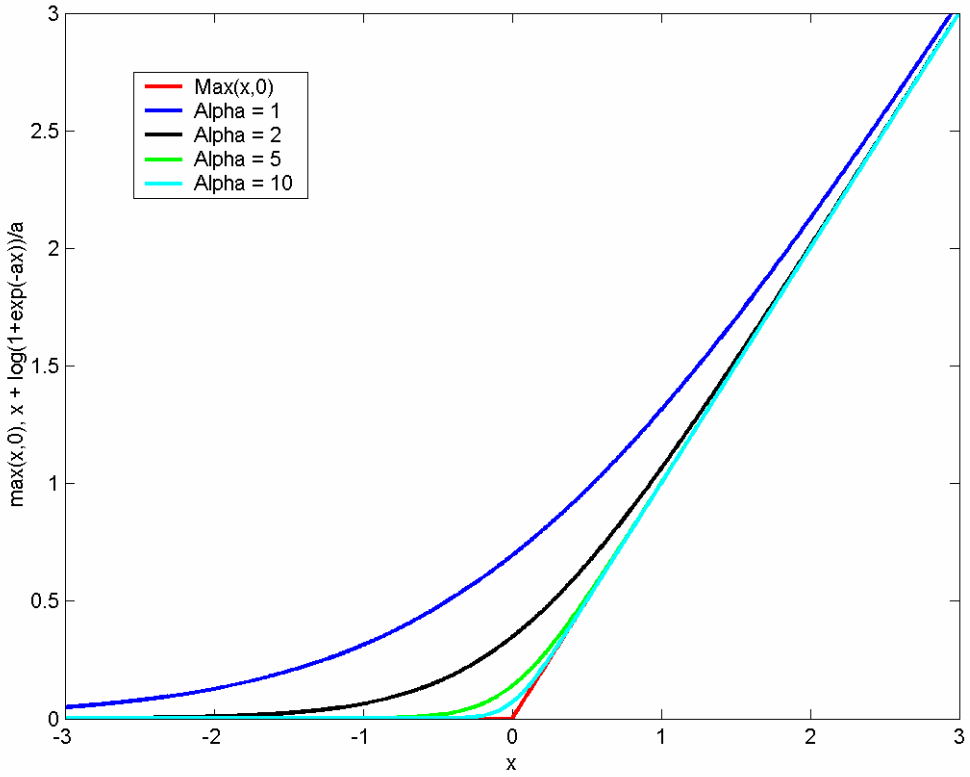

Figure 3 The *max* function and its approximations for various values of α .

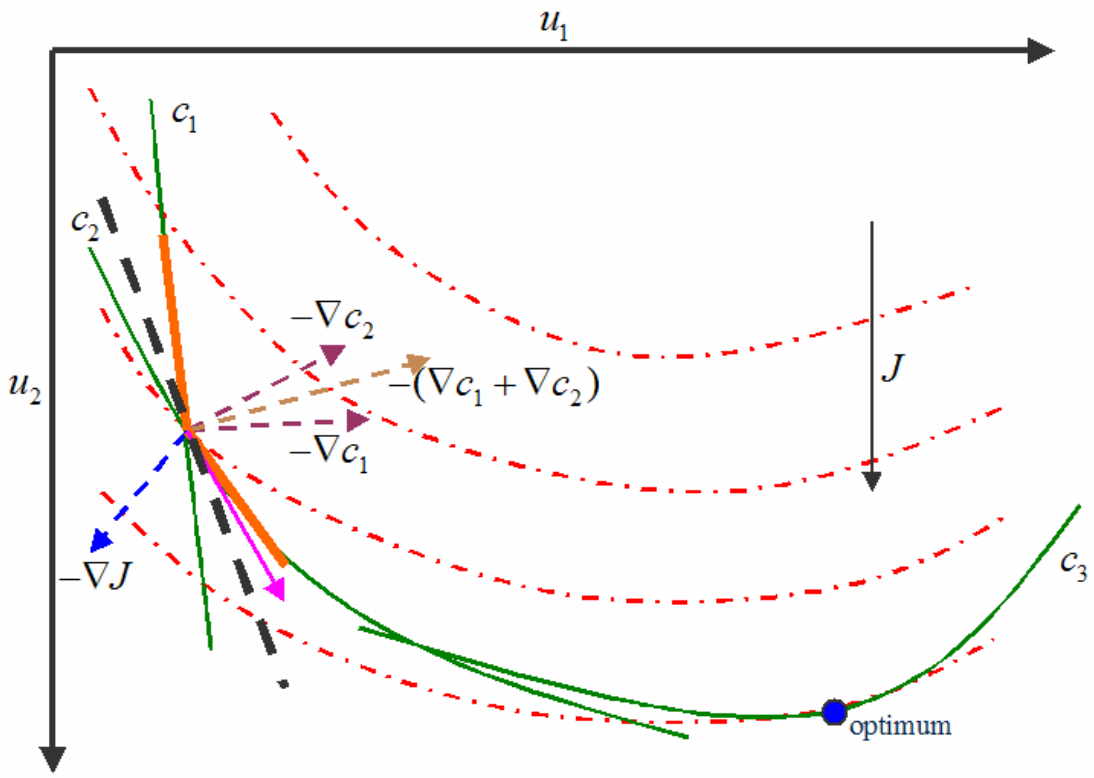

Figure 4 Schematic of a simple optimization problem with constraints, illustrating an approximate feasible direction.

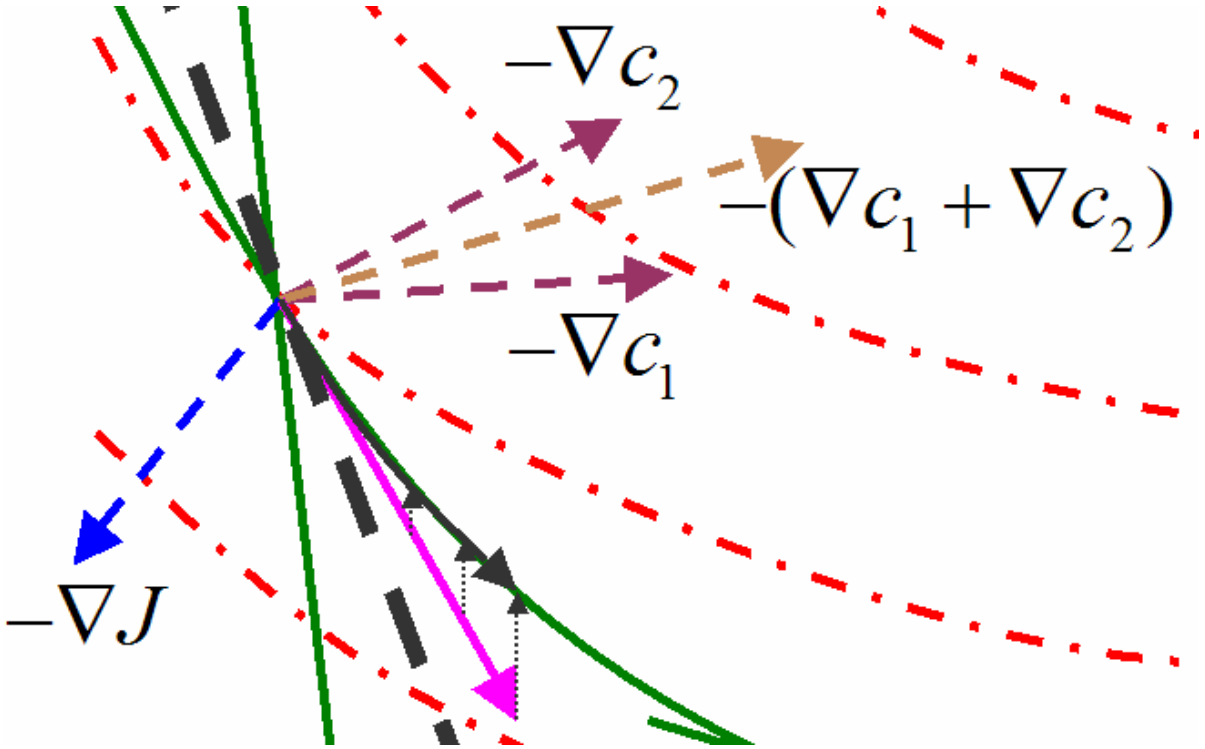

Figure 5 Zoomed in version of the above schematic.

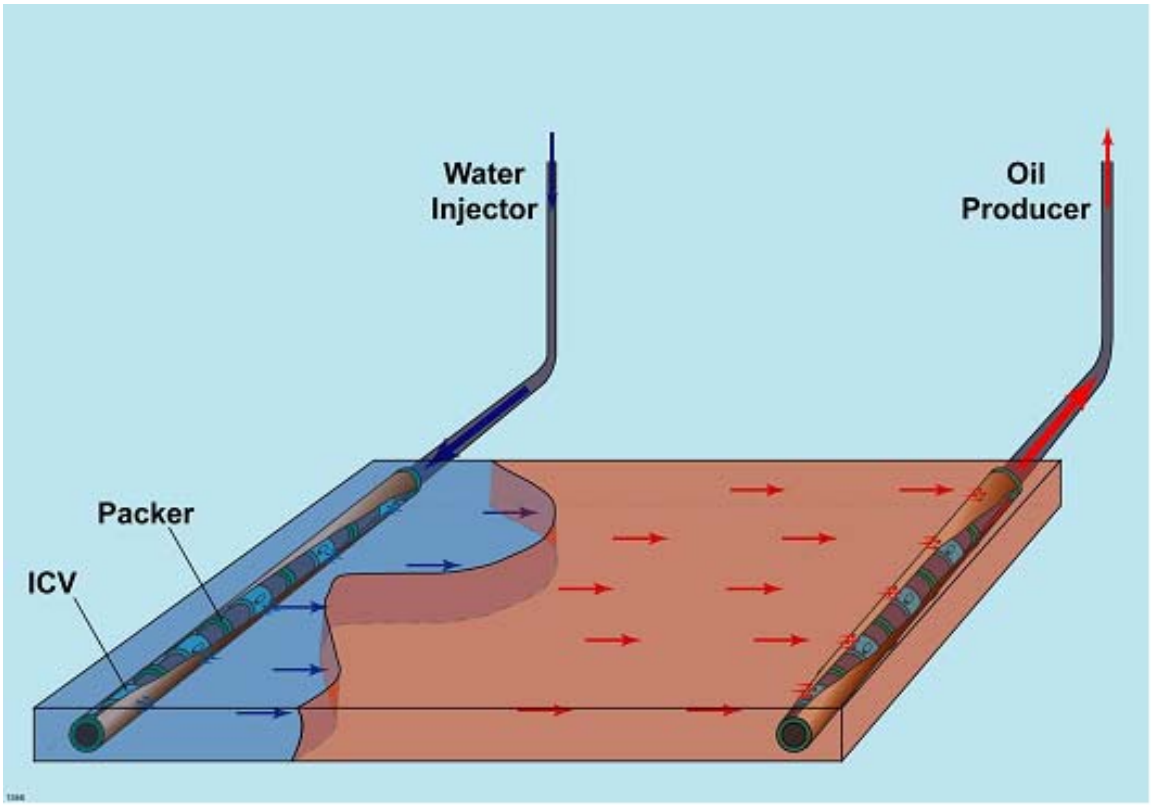

Figure 6 Schematic of reservoir and smart wells for first example (from Brouwer and Jansen [11]).

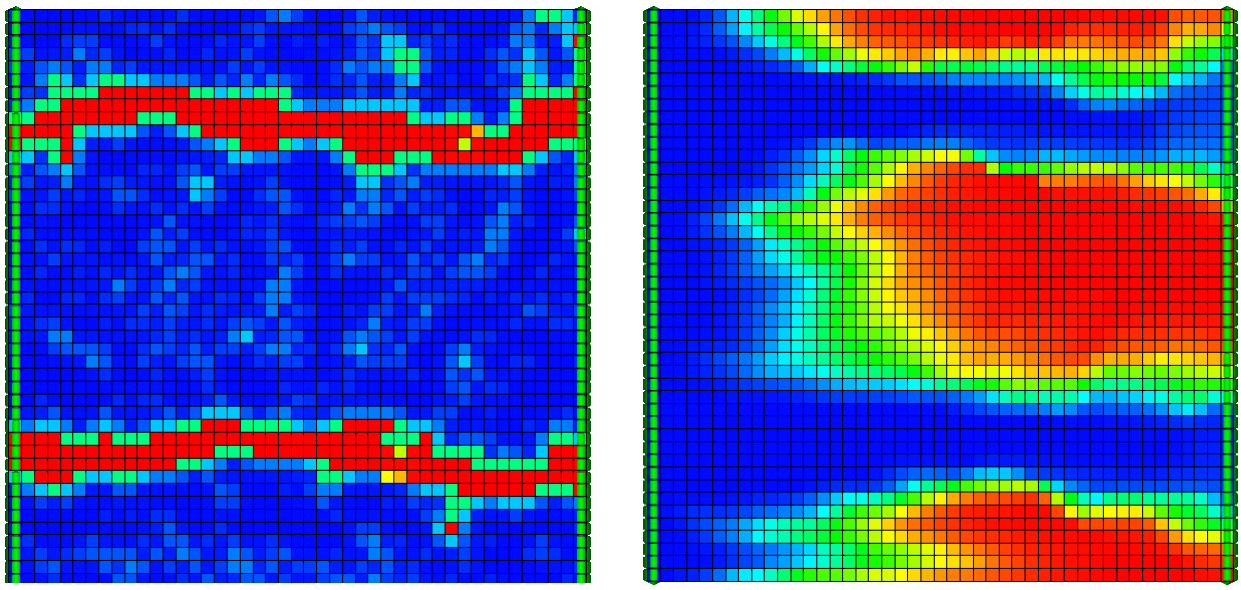

Figure 7 Permeability field for first example on left, and final oil saturation for uncontrolled case on right.

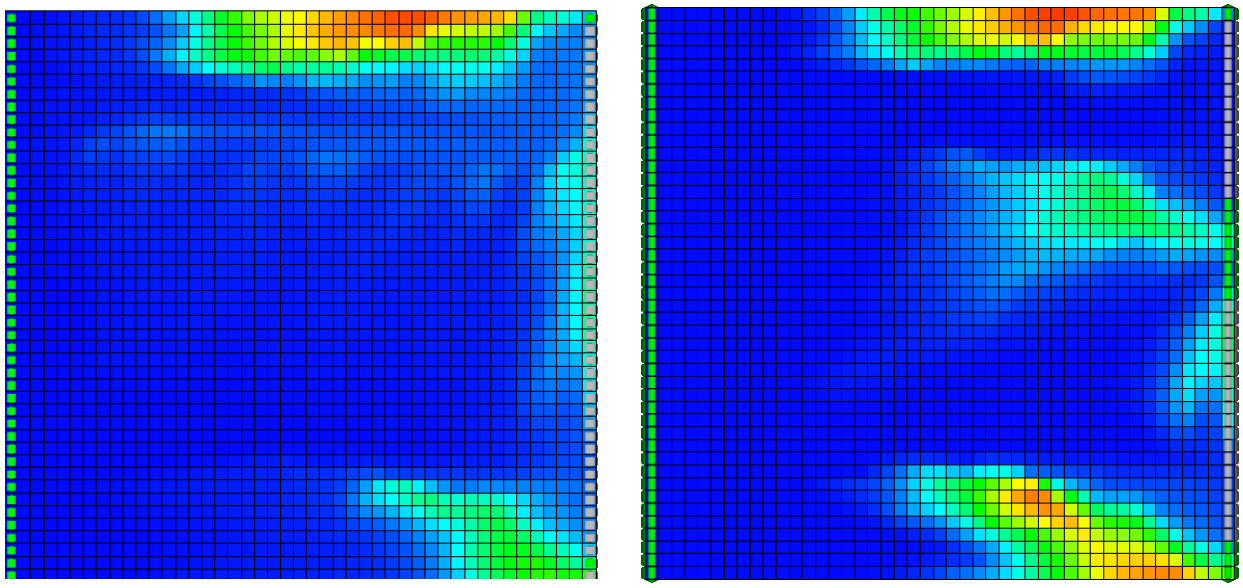

Figure 8 Final oil saturation for rate controlled case on left, and for BHP controlled case on right.

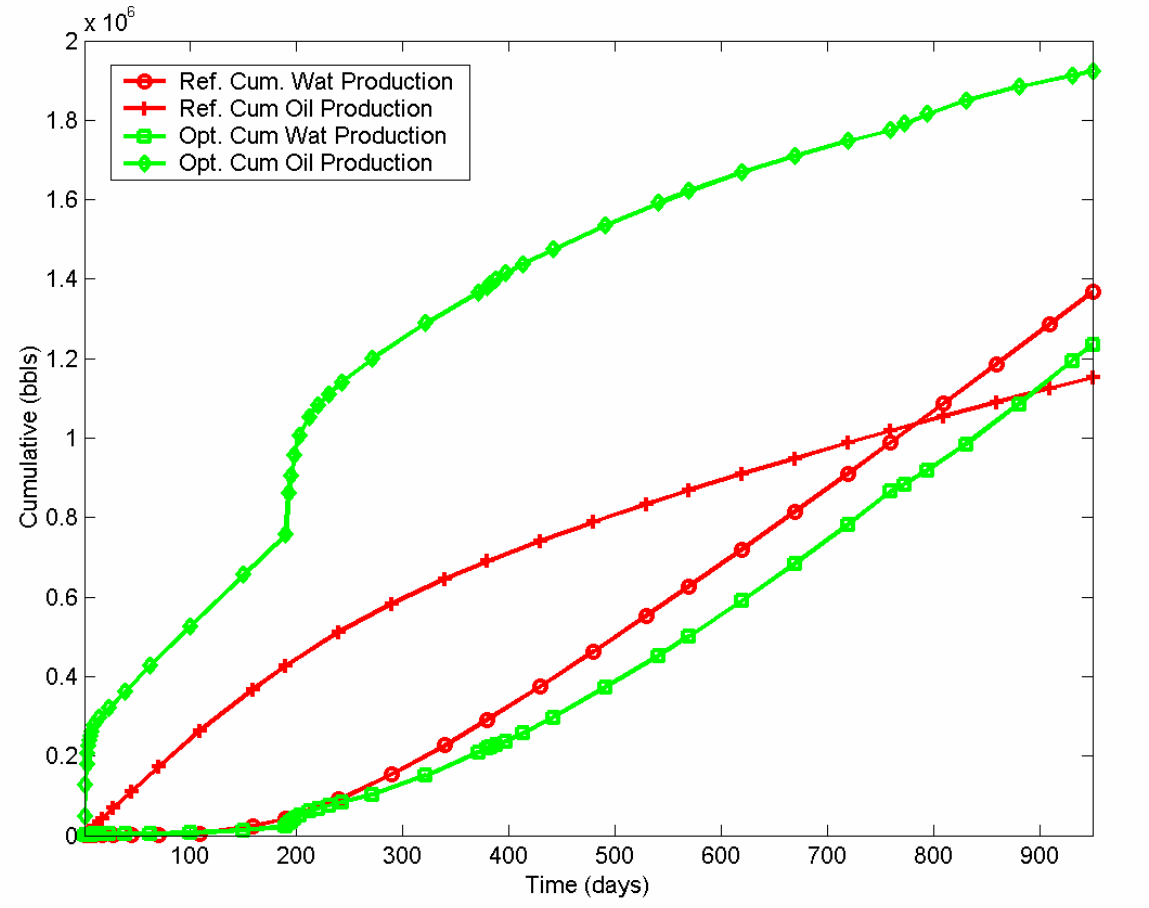

Figure 9 Cumulative water and oil production for uncontrolled reference case and optimized case.

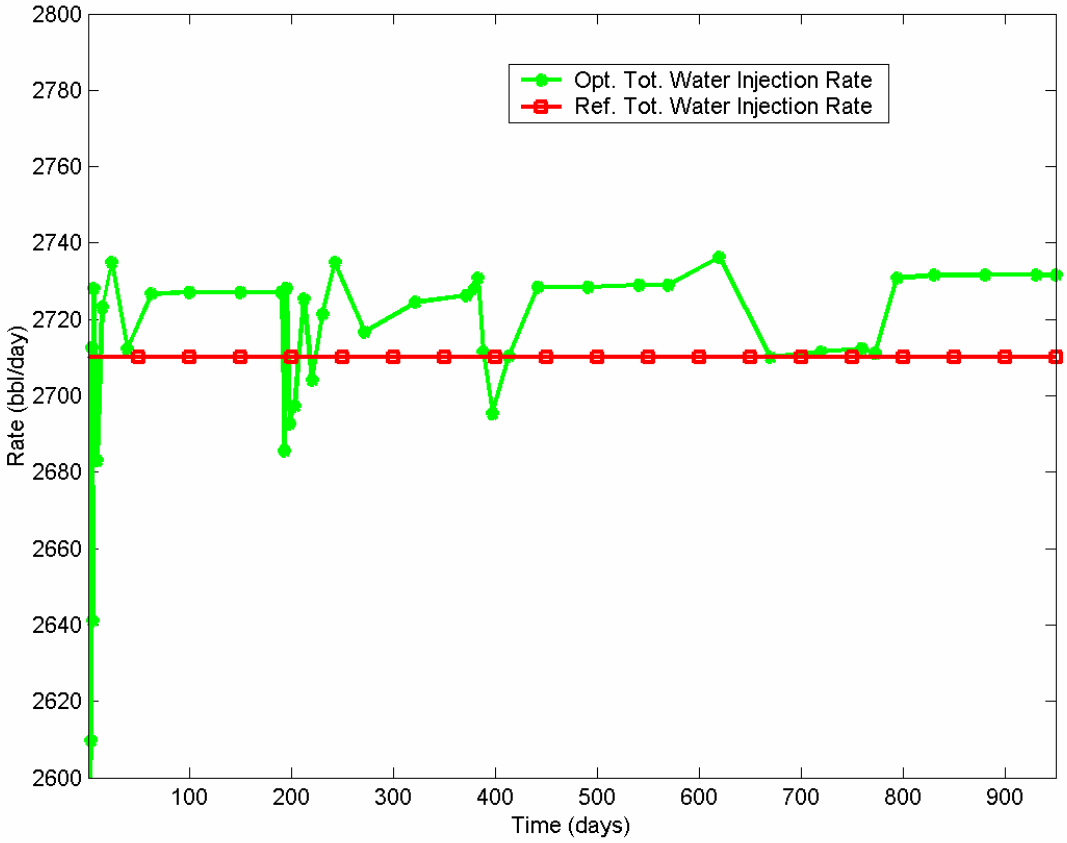

Figure 10 Maximum water injection constraint before and after optimization.

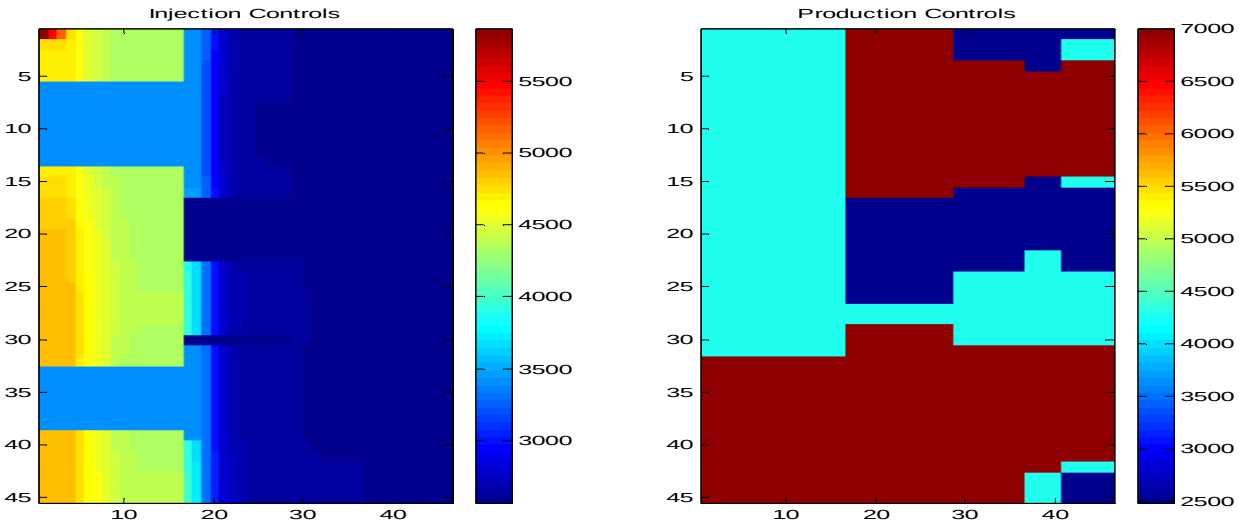

Figure 11 Control trajectories for injectors and producers after optimization.

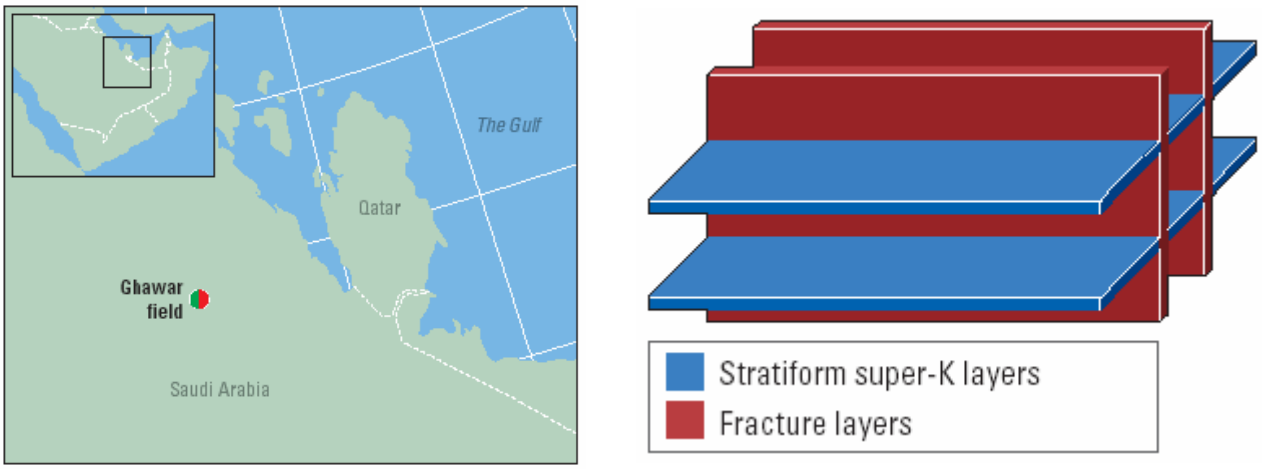

Figure 12 The Ghawar oil field with small rectangle depicting area under study on the left, orientation of fractures and Super – K layers in simulation grid on the right [23].

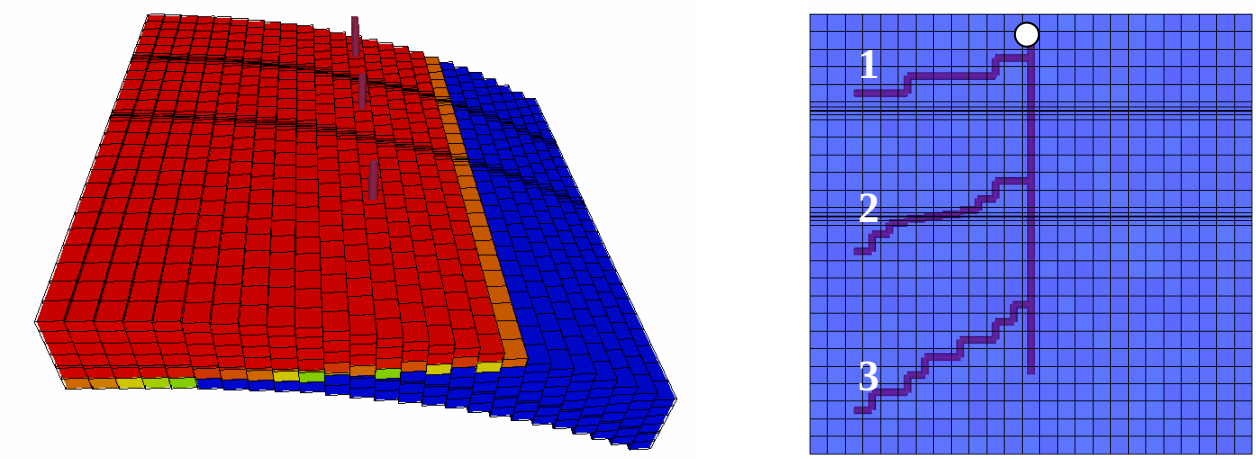

Figure 13 3D simulation model of the Ghawar example on the left, and the tri-lateral well on the right.

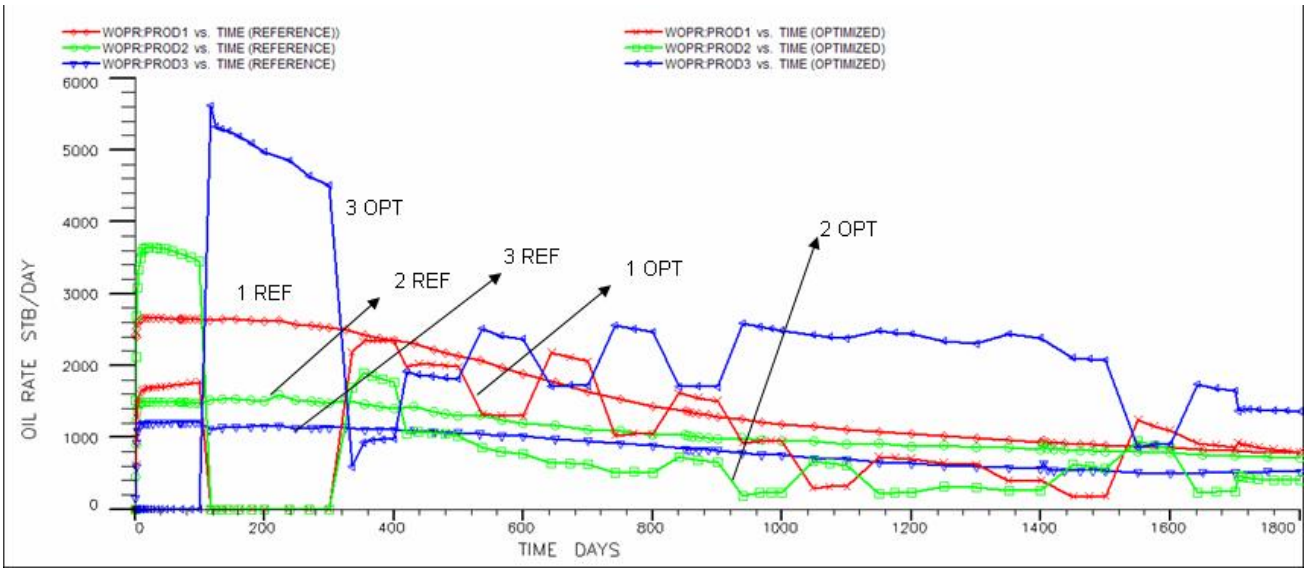

Figure 14 Oil production rates for the three branches for the uncontrolled and optimized cases.

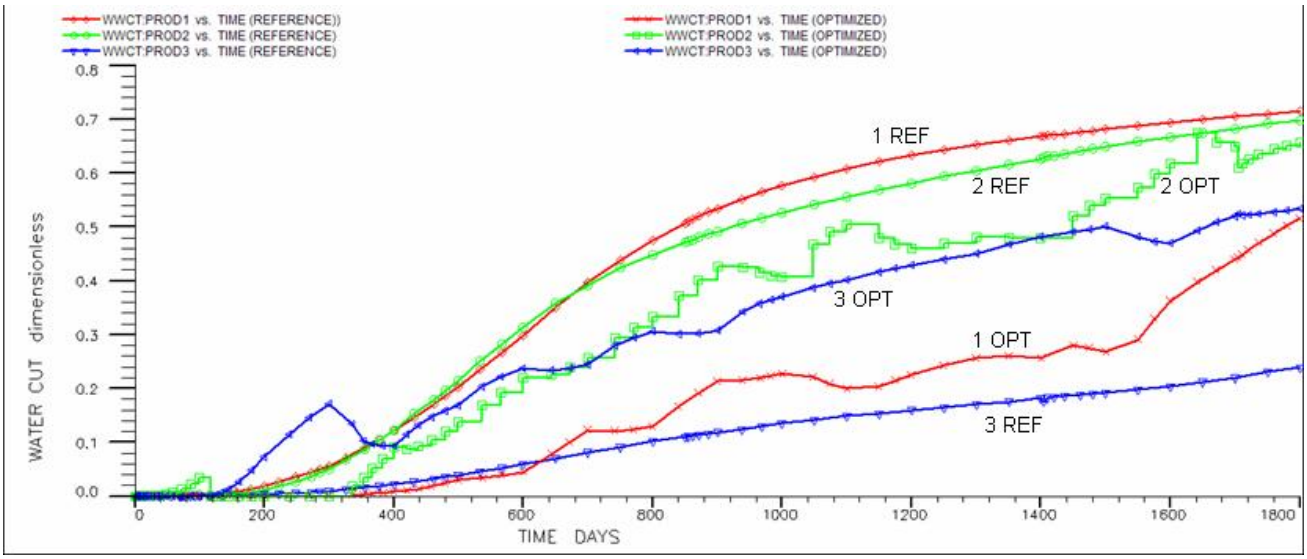

Figure 15 Water cuts for the three branches for the uncontrolled and optimized cases.

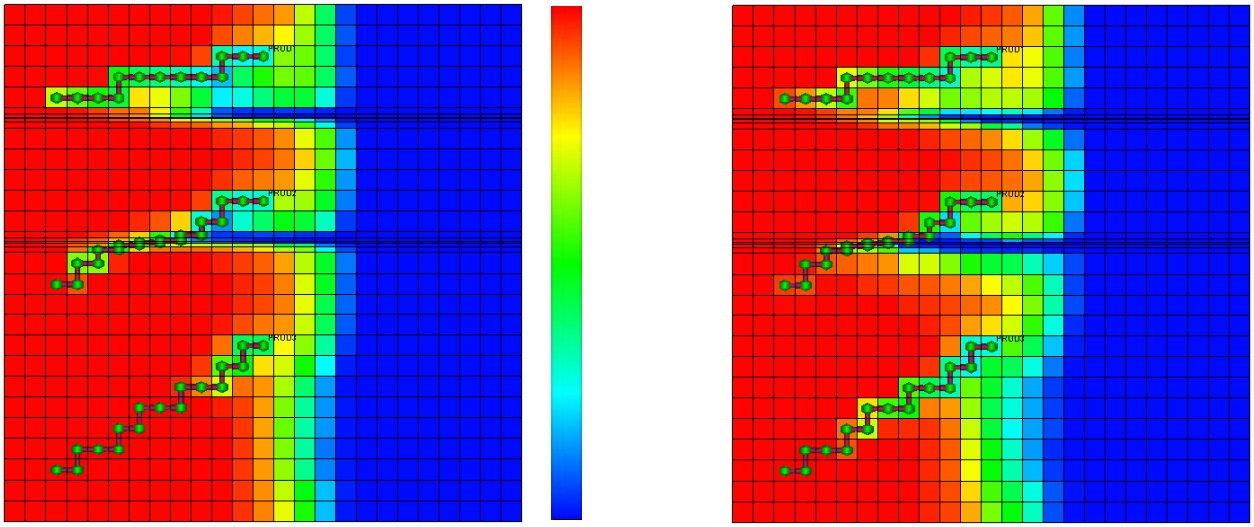

Figure 16 Final oil saturation for layer 2 for the uncontrolled case (left) and the optimized case (right).

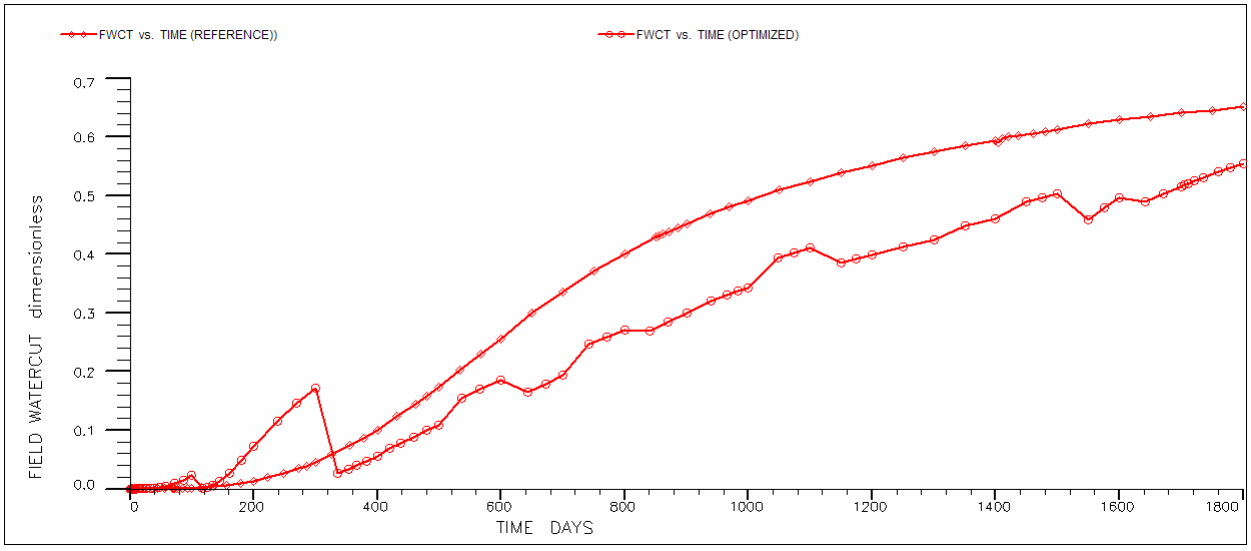

Figure 17 Field water cut for the uncontrolled and optimized cases.

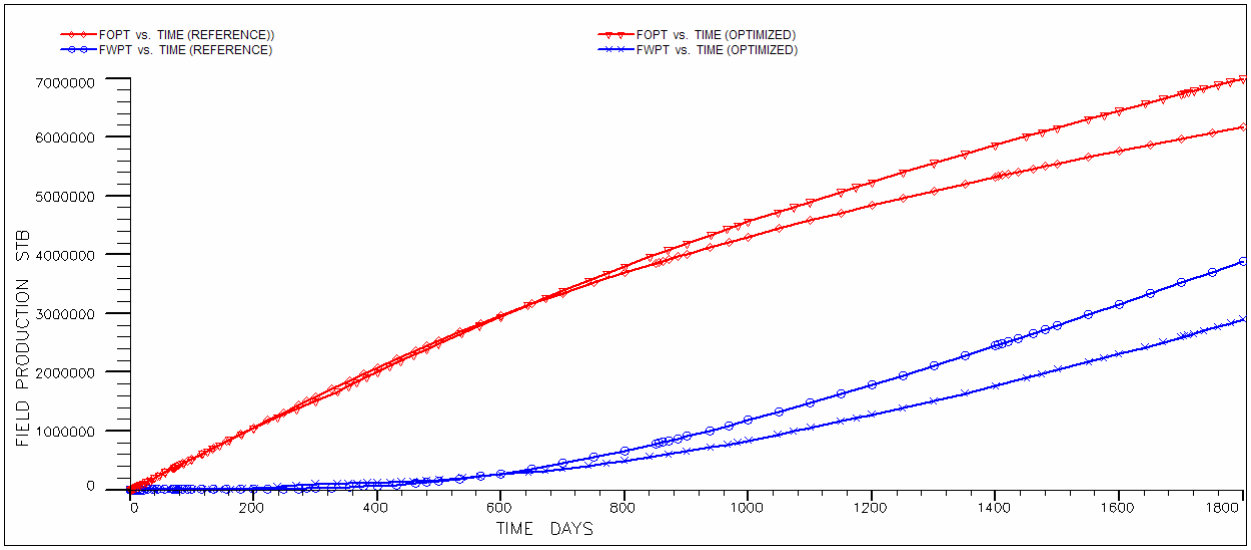

Figure 18 Cumulative oil and water production for the uncontrolled and optimized cases.

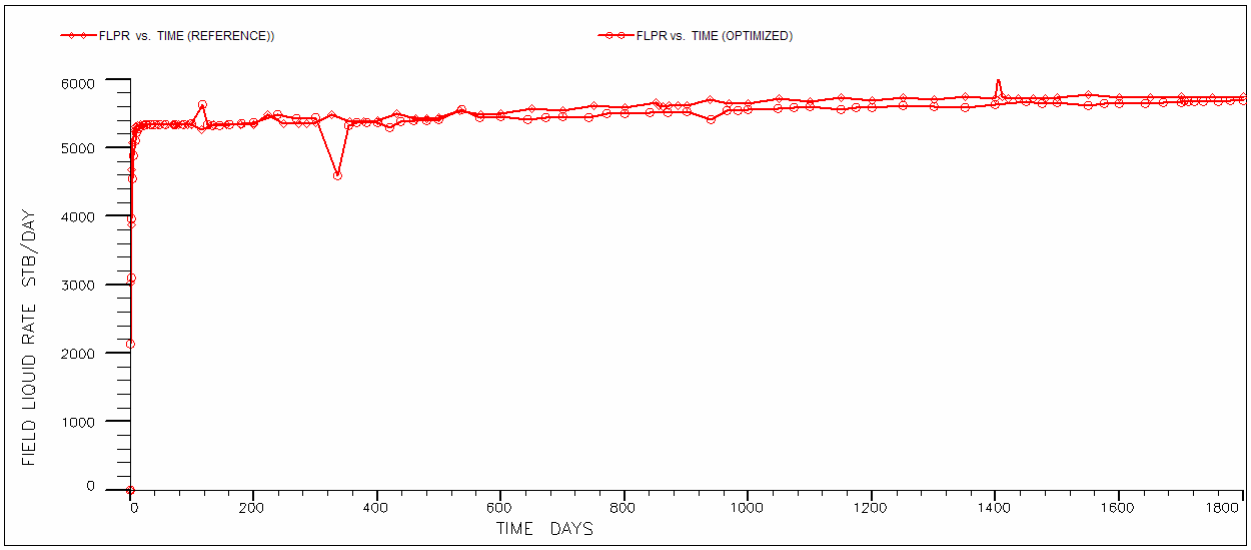

Figure 19 Maximum liquid production constraint for the uncontrolled and optimized cases.