CHAPTER 25

APPROXIMATE DYNAMIC PROGRAMMING FOR OPTIMIZING OIL PRODUCTION

ZHENG WEN, LOUIS J. DURLOFSKY, BENJAMIN VAN ROY
AND KHALID AZIZ

Stanford University, Stanford, California, 94305, USA

Abstract

In this chapter, a new ADP algorithm integrating (1) systematic basis function construction, (2) a linear programming (LP) approach in DP, (3) adaptive basis function selection and (4) bootstrapping, is developed and applied to oil production problems. The procedure requires the solution of a large-scale dynamic system, which is accomplished using a subsurface flow simulator, for function evaluations. Optimization results are presented for cases involving single-phase primary oil production and water injection. In the first case the global optimum can be computed, and the ADP results are shown to essentially achieve this optimum. Clear improvement, relative to various baseline strategies, is similarly observed in the second case. Components

of the new algorithm, or even the entire procedure, may also be applicable in other domains.

25.1 INTRODUCTION

Recently there has been increasing interest in applying advanced computational optimization techniques to improve the performance of petroleum reservoir operations. Many optimization algorithms have been applied to maximize reservoir performance. Most of these optimization algorithms can be classified into two categories: gradient-based/direct-search (e.g., [5, 21, 12]) and global stochastic search (e.g., [16, 9, 1]). Both classes of optimization algorithms face limitations: gradient-based and direct-search algorithms settle for local optima, while global stochastic search algorithms such as genetic algorithms typically require many function evaluations for convergence, and even then there is no assurance that the global optimum has been found.

As we will see in Section 25.2, the petroleum reservoir production optimization problem can be formulated as a dynamic optimal control problem. Thus, in principle, the globally optimal control (global optimum) can be computed via dynamic programming (DP) [3]. However, most reservoir production problems of practical interest are large-scale, nonlinear dynamic optimization problems. Thus, except for a few special cases, if we directly apply DP to such problems, the computational requirements become prohibitive due to the curse of dimensionality. In particular, time and memory requirements typically grow exponentially with the number of state variables.

One classical way to overcome the curse of dimensionality in DP is to apply approximate dynamic programming (ADP), which aims to address this computational burden by efficiently approximating the value function (see [4, 26, 20] for more on ADP). The result is an approximate value function, which is typically represented by a linear combination of a set of predefined basis functions. ADP algorithms provide methods for computing the coefficients associated with these basis functions. ADP has been successfully applied across a broad range of domains such as asset pricing

[24, 19, 25], transportation logistics [22], revenue management [28, 13, 29], portfolio management [15], and even to games such as backgammon [23].

In this chapter, we propose a new ADP algorithm based on the linear programming (LP) approach and apply it to two reservoir production problems. We also compare the performance of ADP with that achieved using various baseline strategies. As we will show, ADP performs well relative to the baseline results for the problems considered.

Our contribution in this work is twofold: from the perspective of reservoir production problems, in addition to proposing a new approach that achieves promising performance, our ADP algorithm offers some advantages relative to alternative optimization techniques. For example, ADP can readily handle general nonlinear constraints in reservoir production, such as enforcing a maximum production rate for water (which is often produced along with oil), which may not be straightforward to handle using alternative algorithms. In addition, the ADP algorithm we propose is essentially deterministic, yet it aims to approximate a global optimum. This distinguishes it from stochastic global search algorithms that have been applied to oil production optimization.

Our proposed ADP algorithm combines four key ingredients: (1) systematic basis function construction based on proper orthogonal decomposition (POD), (2) coefficient computation based on smoothed reduced linear programming (SRLP), (3) selection of basis functions via an adaptive procedure, and (4) performance enhancement through bootstrapping. To our knowledge, both POD-based basis function construction and our adaptive basis function selection method are new. In addition, this work represents the first attempt to incorporate all four of these techniques together and apply them to large-scale dynamic optimization problems. The simulation results indicate that the overall approach is promising, and we expect that the ideas developed here are also applicable in other domains.

The remainder of this chapter is organized as follows. In Section 25.2, we introduce the reservoir production optimization problem. The basic DP and ADP procedures, especially the linear programming approach, are reviewed in Section 25.3. Then, in Section 25.4, we propose our ADP algorithm and discuss its computational demands.

Simulation results for linear and nonlinear examples are presented in Section 25.5. We conclude this contribution in Section 25.6.

The basic formulation appearing in this chapter was presented previously (in a Society of Petroleum Engineers conference paper, [27]) and the current description closely follows the earlier presentation.[1] The examples presented in Section 25.5 are, however, different from (and more realistic than) those in the earlier work.

25.2 PETROLEUM RESERVOIR PRODUCTION OPTIMIZATION PROBLEM

In this section, we introduce the petroleum reservoir production optimization problem and formulate it as an optimal control problem. Our treatment here is brief – interested readers can refer to [2, 16, 5, 21, 7, 12] for detailed explanations of petroleum reservoir simulation and production optimization.

Reservoir fluids, such as oil and water, reside within the pore space of porous rock formations. Oil can be produced via primary production, in which reservoir fluid is removed but no fluid is injected, as well as via water injection. In the latter case, injected water maintains reservoir pressure and sweeps oil to production wells. In order to model subsurface flow systems such as oil reservoirs, we must specify a geological model (which defines rock properties and the flow domain), fluid and rock-fluid properties, and well locations and controls, among other parameters. The key rock property that impacts flow is permeability, which relates flow rate to pressure gradient (permeability can thus be viewed as a flow conductivity). Flow dynamics are described by one or more (depending on the number of components being tracked) partial differential equations that combine statements of mass conservation with constitutive relations.

In practical applications, the governing equations are discretized on a grid and a numerical solution technique is applied (finite-volume procedures are typically used

[1] SPE holds the copyright of the problem formulation and algorithm description portions of this chapter (Sections 25.2–25.4), which are reproduced with the permission of the copyright owner. Further reproduction is prohibited without permission.

in reservoir simulation). Following spatial discretization, these equations can be represented as a system of ordinary differential equations (ODEs):

$$d\mathbf{x}(t)/dt = F\left(\mathbf{x}(t), \mathbf{u}(t)\right), \tag{25.1}$$

subject to an initial condition $\mathbf{x}(0) = \mathbf{x}_0$ and instantaneous constraints $S(\mathbf{x}(t), \mathbf{u}(t)) \leq 0$. Here $\mathbf{x}(t)$ and $\mathbf{u}(t)$ denote the reservoir states and the control action at time t, respectively. Typically, $\mathbf{x}$ includes the grid-block pressure and saturation values (saturation is the volume fraction of a particular phase in the grid block) and $\mathbf{u}$ encodes the well settings (well pressures in the reservoir, referred to as bottomhole pressures or BHPs, and/or well flow rates) of the injection and production wells. The constraints $S(\mathbf{x}(t), \mathbf{u}(t)) \leq 0$ restrict the state and control action at time t. Constraints in reservoir production problems typically include upper/lower bounds on BHPs and upper/lower bounds on flow rates of fluids (e.g., maximum water rate, minimum oil rate).

We also assume that at time t, a payoff accumulates at a rate given by an instantaneous payoff function $L(\mathbf{x}(t), \mathbf{u}(t))$ that depends on the state $\mathbf{x}(t)$ and control action $\mathbf{u}(t)$. A widely used instantaneous payoff function is

$$\begin{aligned} L(\mathbf{x}(t), \mathbf{u}(t)) \quad &= \quad \text{revenue from producing oil} \\ &- \quad (\text{cost for producing water} + \text{cost for injecting water})\,. \end{aligned} \tag{25.2}$$

In our optimizations, we aim to maximize the net present value (NPV) of the cumulative payoff $\int_0^T e^{-\alpha t} L\left(\mathbf{x}(t), \mathbf{u}(t)\right) dt$ by determining the optimal well settings (BHPs in this case) over a time horizon $[0, T]$, where $\alpha \geq 0$ is the continuous-time discount rate that captures the time value of the payoff.

In most oil production optimization problems, the termination time T is large enough so that the difference between the NPV of the cumulative profit

$$\int_0^T e^{-\alpha t} L\left(\mathbf{x}(t), \mathbf{u}(t)\right) dt,$$

and its infinite horizon approximation

$$\int_0^\infty e^{-\alpha t} L\left(\mathbf{x}(t), \mathbf{u}(t)\right) dt,$$

is sufficiently small. Moreover, it is well known that there exists a stationary policy $\mathbf{u}^*(t) = \mu^*(\mathbf{x}(t))$ that maximizes the infinite horizon discounted objective. As we will see later, the existence of a stationary globally optimal policy μ^* simplifies our ADP algorithms. For this reason, we will focus on solving the infinite horizon dynamic optimization problem:

$$\begin{aligned} \max_{\{u(t),t\geq 0\}} \quad & \int_{t=0}^{\infty} e^{-\alpha t} L(\mathbf{x}(t),\mathbf{u}(t))dt \\ \text{s.t.} \quad & d\mathbf{x}(t)/dt = F(\mathbf{x}(t),\mathbf{u}(t)) \\ & \mathbf{x}(0) = \mathbf{x}_0 \text{ and } S(\mathbf{x}(t),\mathbf{u}(t)) \leq \mathbf{0}. \end{aligned} \tag{25.3}$$

The optimization problem (25.3) for practical reservoir production problems is very challenging for the following reasons. First, for two-phase or multiphase flow cases, F is a highly nonlinear function, so the ODE (25.1) is nonlinear. Second, the dimension of the state space is very high. Specifically, $\mathbf{x} \in \Re^{N_b N_p}$, where N_b is the number of grid blocks used to discretize the system and N_p is the number of components. For instance, for a case with $40 \times 40 \times 5$ grid blocks and two components (oil and water), the dimension of the state space is $16,000$.

The performance (NPV of the cumulative payoff) of any control strategy (policy) μ is evaluated through numerical solution of the governing equations (i.e., by performing a reservoir simulation). As described in Subsection 25.4.2, reservoir simulation is also used for constraint sampling. Since most of the computation of our proposed ADP algorithm is consumed by reservoir simulations, it is appropriate to measure the computational demands of the algorithm in terms of the number of reservoir simulations that must be performed. This issue is discussed in detail in Subsection 25.4.5.

25.3 REVIEW OF DYNAMIC PROGRAMMING AND APPROXIMATE DYNAMIC PROGRAMMING

In this section, we briefly review dynamic programming (DP) and approximate dynamic programming (ADP), especially the linear programming (LP) approach. Dynamic programming (DP) offers a class of optimization algorithms that decompose

a dynamic optimization problem into a sequence of simpler sub-problems. At each time t, an optimal control action $\mathbf{u}^*(t)$ is computed by solving one sub-problem. These sub-problems are coordinated across time through a value function denoted by J^*. The value function captures the future impact of the current action; the algorithm must balance immediate payoff against future possibilities. In our model, the value function is defined by:

$$\begin{aligned} J^*(\mathbf{x}_0) \quad = \quad & \max_{u(t), t\geq 0} \quad \int_{t=0}^{\infty} e^{-\alpha t} L(\mathbf{x}(t), \mathbf{u}(t)) dt \\ & \text{s.t.} \quad d\mathbf{x}(t)/dt = F\left(\mathbf{x}(t), \mathbf{u}(t)\right), \\ & \quad \mathbf{x}(0) = \mathbf{x}_0 \text{ and } S\left(\mathbf{x}(t), \mathbf{u}(t)\right) \leq \mathbf{0}. \end{aligned} \tag{25.4}$$

Note that $J^*(\mathbf{x}_0)$ is the maximum net present value starting from state $\mathbf{x}_0$ at time 0. Further, given the time homogeneity of our model, the value $J^*(\mathbf{x}(t))$ at any time t and for any state $\mathbf{x}(t)$ is the maximum net present value of payoffs that can be accumulated starting at that time and state.

Given the value function J^* and current state $\mathbf{x}(t)$, an optimal control action at time t can be selected by solving the following optimization problem [3]:

$$\begin{aligned} \max_{\mathbf{u}} \quad & L\left(\mathbf{x}(t), \mathbf{u}\right) + F\left(\mathbf{x}(t), \mathbf{u}\right)^T \nabla J^*\left(\mathbf{x}(t)\right) \\ \text{s.t.} \quad & S(\mathbf{x}(t), \mathbf{u}) \leq \mathbf{0}, \end{aligned} \tag{25.5}$$

where ∇J^* is the gradient of J^*. Note that this problem decouples the choice of control action at time t from that at all other times. It is in this sense that DP decomposes the dynamic optimization problem into simpler sub-problems through use of the value function. As we will see later, for reservoir production problems the sub-problem (25.5) can be solved efficiently.

In light of the relative ease of generating optimal control actions given the value function, the challenge in optimizing reservoir production using DP (or ADP) is in the computation of the value function. As discussed in [15], the optimal value function for a continuous-time optimal control problem can in principle be computed by solving the Hamilton-Jacobi-Bellman (HJB) equation:

$$\max_{\mathbf{u}:\, S(\mathbf{x},\mathbf{u})\leq \mathbf{0}} \left\{ L(\mathbf{x}, \mathbf{u}) + F(\mathbf{x}, \mathbf{u})^T \nabla J(\mathbf{x}) - \alpha J(\mathbf{x}) \right\} = 0, \tag{25.6}$$

which is a partial differential equation.

An alternative though closely related characterization of the value function is an infinite dimensional linear program (LP):

$$\begin{aligned} \min_{J \in C^2} \quad & \int J(\mathbf{x}) \pi(d\mathbf{x}) \\ \text{s.t.} \quad & L(\mathbf{x}, \mathbf{u}) + F(\mathbf{x}, \mathbf{u})^T \nabla J(\mathbf{x}) - \alpha J(\mathbf{x}) \leq 0 \quad \forall \mathbf{x}, \mathbf{u} \text{ s.t. } S(\mathbf{x}, \mathbf{u}) \leq \mathbf{0}, \end{aligned} \tag{25.7}$$

where π is a probability measure chosen with exhaustive support such that the integral of J^* is finite. This LP poses an infinite number of decision variables and an infinite number of constraints since there is one decision variable $J(\mathbf{x})$ per state $\mathbf{x}$ and there is one constraint $S(\mathbf{x}, \mathbf{u}) \leq 0$ per state-action pair $(\mathbf{x}, \mathbf{u})$.

From Theorem 1.1.3 of [15], we have:

Theorem 1: *If $J^* \in C^2$, then J^* uniquely attains the optimum in (25.7).*

For a given reservoir production problem, it is not clear whether or not the value function J^* satisfies the technical condition $J^* \in C^2$. However, as will be described in Section 25.4, the approximate value function $\tilde{J}$ in our proposed ADP algorithm does satisfy $\tilde{J} \in C^2$. As a result, the technical condition $J^* \in C^2$ does not impose any additional restriction for our proposed ADP algorithm.

Similar to other large-scale optimization problems, in most petroleum reservoir production problems of practical interest, solving the HJB equation (25.6) or the infinite dimensional LP (25.7) exactly is impossible, as this would require computing and storing J^* for each state in a continuous state space. Even if we were to discretize the state space by quantizing each state variable into multiple discrete values, the number of discrete states would grow exponentially with the dimension of the state space D ($D = N_p N_b$), again making storage and computation impractical.

In a few special cases, the value function assumes a special structure which facilitates efficient computation and storage. This is the case, for example, with single-phase reservoir models with no constraints on control actions and an instantaneous payoff function that is quadratic in state and control action. In this context, the value function is itself quadratic and the problem reduces to one of linear-quadratic control (LQ). Such problems are well known to be tractable. For more realistic prob-

lems, involving for example two-phase flow, the reservoir dynamics are nonlinear and computing the value function exactly is infeasible.

One way to overcome the curse of dimensionality is to approximate the value function J^*. Deriving a near-optimal control strategy $\tilde{\mu}$ through appropriate value function approximation is known as approximate dynamic programming (ADP). As is common in ADP, we will approximate the value function of our dynamic optimization model in terms of a linear combination of a selected set of basis functions:

$$J^*(\mathbf{x}) \approx \tilde{J}(\mathbf{x}) = \sum_{k=0}^{K} r_k \phi_k(\mathbf{x}), \tag{25.8}$$

where $\phi_k(\cdot), k = 0, \cdots, K$ is a set of basis functions, and the r_k's are their respective coefficients. As we will see, the main advantage of the linear parameterization in the approximation structure (25.8) is that various "approximate" versions of the LP (25.7) (e.g., ALP (25.9), RLP (25.10) and SRLP (25.11)) are themselves linear programming problems and hence can be solved efficiently if they have finite constraints and decision variables. Replacing the exact value function J^* with the approximation $\tilde{J}$ when solving each sub-problem (25.5) results in a control strategy $\tilde{\mathbf{u}} = \tilde{\mu}(\mathbf{x})$. We expect that if $\tilde{J}$ is "close" to J^*, the use of controls $\tilde{\mu}$ will provide performance that is "close" to the global optimum.

With the approximation structure (25.8), we only need to specify how to construct basis function ϕ_k's and compute their coefficients (weights) r_k's. In the next section, we will describe how we implement our ADP algorithm and discuss some techniques to further improve its performance.

25.4 APPROXIMATE DYNAMIC PROGRAMMING ALGORITHM FOR RESERVOIR PRODUCTION OPTIMIZATION

In this section, we develop a new ADP algorithm based on the linear programming (LP) approach to compute a near-optimal control for the petroleum reservoir production problem presented in Section 25.2. Our ADP algorithm proceeds as follows. We first construct a pool of candidate basis functions $\phi_k(\cdot)$, where $k = 0, 1, \cdots, \tilde{K}$. Next, we apply adaptive basis function selection to effectively choose basis functions

from the candidate basis function pool. Third, we compute the coefficients (weights) r_k of the selected basis functions. We then derive near-optimal control actions $\tilde{\mathbf{u}}$ using the approximate value function $\tilde{J}$ by solving sub-problem (25.5). Finally, we use bootstrapping to further enhance the performance of the ADP algorithm.

In the remainder of this section, we will discuss our implementation of each of these key steps. The overall procedure is illustrated in Figure 25.1.

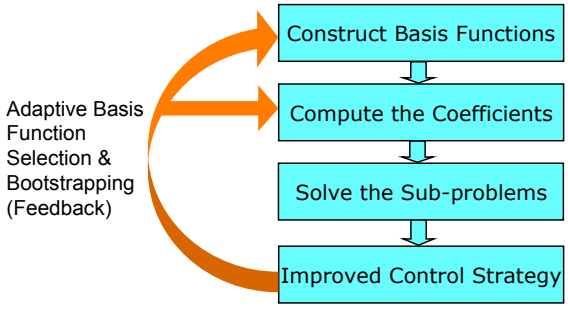


Figure 25.1: Flowchart of proposed ADP algorithm for oil production optimization.

25.4.1 Basis Function Construction

In this subsection, we describe the construction of the candidate basis functions $\phi_k(\cdot)$. Although there are studies focusing on constructing basis functions automatically (e.g., [18]), in most applications, in order to achieve a more accurate approximation of the value function, problem-dependent approaches are used. For oil production problems, we propose an approach to construct the candidate basis functions using proper orthogonal decomposition (POD). The candidate basis function pool is built using these basis functions as well as some other pre-specified basis functions.

In our approach, the value function J^* is the net present value of the future total profit under the optimal control strategy. Hence, the basis functions should encode information about aspects of the states that strongly impact future profit. We have observed that the following two categories of functions are correlated to future profit, which has motivated us to choose basis functions accordingly:

1. Functions reflecting the "global status" of the reservoir, such as the average oil and water saturations and the average pressure. As we have discussed above, a systematic way to capture the global status is through proper orthogonal decomposition (POD) of global snapshots. We will discuss how to construct basis functions based on POD later.

2. Functions reflecting the status of the near-well regions, such as the average oil saturation near production wells, average pressure near injection wells, etc.

In addition to the values described above, it is also useful to include higher-order polynomial functions of these values, as discussed below.

POD identifies a low-dimensional subspace that captures most of the system variability. It is used in model-order reduction for petroleum reservoir simulation (e.g., [7]) and other application areas, so it seems reasonable to apply POD for the construction of ADP basis functions. Our use of POD for basis construction proceeds as follows.

We first run the reservoir simulator under a prescribed baseline strategy μ_0 and record "snapshots" of the states. That is, we use the reservoir simulator to solve $d\mathbf{x}/dt = F(\mathbf{x}, \mu_0(\mathbf{x}))$, and sample the state trajectory $\mathbf{x}(t)$ at $t = t_0, t_1, t_2, \cdots, t_{L-1}$. We define $\mathbf{x}_p(t)$ and $\mathbf{x}_S(t)$ to be vectors containing the pressure and saturation components of $\mathbf{x}(t)$, respectively (for an oil-water problem). The sample means are denoted by $\bar{\mathbf{x}}_p$ and $\bar{\mathbf{x}}_s$.

The centered snapshot matrices for pressure and saturation are

$$\begin{aligned} X_p &= [\mathbf{x}_p(t_0) - \bar{\mathbf{x}}_p, \mathbf{x}_p(t_1) - \bar{\mathbf{x}}_p, \cdots, \mathbf{x}_p(t_{L-1}) - \bar{\mathbf{x}}_p] \\ X_S &= [\mathbf{x}_S(t_0) - \bar{\mathbf{x}}_S, \mathbf{x}_S(t_1) - \bar{\mathbf{x}}_S, \cdots, \mathbf{x}_S(t_{L-1}) - \bar{\mathbf{x}}_S], \end{aligned}$$

respectively. The subspace can now be characterized using singular value decomposition (SVD). Specifically, after applying SVD, we decompose X_p and X_S as $X_p = U_p \Sigma_p V_p^T$ and $X_S = U_S \Sigma_S V_S^T$, respectively, where Σ_p and Σ_S are diagonal matrices made up of singular values in descending order. Let $U_p(1 : N_p)$ and $U_S(1 : N_S)$ denote the first N_p and N_S columns of U_p and U_S, respectively. We

define the projection matrix

$$\mathbf{\Phi} = \begin{bmatrix} U_p(1:N_p) & \mathbf{0} \\ \mathbf{0} & U_S(1:N_S) \end{bmatrix}.$$

Note that span($\mathbf{\Phi}$) is a subspace that captures most of the variation in pressure and saturation.

Now, we construct basis functions that are polynomials defined on span($\mathbf{\Phi}$). Specifically, letting φ_i denote the ith column of $\mathbf{\Phi}$, the basis functions take the form $\phi_k(\mathbf{x}) = M_k \prod_{i=1}^{I} \left(\varphi_{k_i}^T \mathbf{x}\right)^{m_i}$, where m_i are nonnegative integers, M_k is a normalization constant and I is the number of terms in the basis function ϕ_k. We say ϕ_k is a one-term polynomial if $I = 1$.

This approach is based on the assumption that the reservoir dynamics under the baseline strategy μ_0 are sufficiently similar to those under the optimal strategy μ^*. This is not always the case. One way to address this issue involves bootstrapping, as we will describe later.

It is evident that we can potentially construct many (polynomial) functions reflecting either the status of the entire reservoir or the status of near-well regions. Thus, there are many candidate basis functions. However, the use of too many basis functions can result in onerous computational requirements or performance loss due to overfitting. Thus, in practice, only a subset of all of these candidate basis functions are used to approximate the value function. We will later propose an adaptive basis function selection scheme, which chooses an effective subset of basis functions.

Furthermore, no matter how we construct and choose basis functions, we always include a constant basis function $\phi_0(\mathbf{x}) = 1$ to achieve better approximation results and numerical stability. We also normalize all basis functions such that $|\phi_k(\mathbf{x}_0)| = 1$ for $k = 0, \cdots, K$, where $\mathbf{x}_0$ is the initial state.

Finally, we observe that any candidate basis function ϕ_k constructed as described above satisfies the technical condition $\phi_k \in C^2$. Since the approximate value function $\tilde{J}$ is a linear combination of a selected set of candidate basis functions, $\tilde{J} \in C^2$.

25.4.2 Computation of Coefficients

As discussed above, for a given set of basis functions, we need to compute coefficients to approximate the value function. A number of ADP algorithms have been proposed for this purpose. Examples include approximate value iteration, approximate policy iteration, temporal-difference learning, Bellman error minimization and approximate linear programming (ALP) (see [4, 26, 20]). In this chapter, we use smoothed reduced linear programming (SRLP) with L_1 regularization. This is a variant of ALP proposed in [11]. Compared with other ADP algorithms, the main advantage of this procedure (in our context) is that it provides "optimal" solutions in much less time than alternative approaches. In the remainder of this section, we describe the SRLP algorithm with L_1 regularization. An important aspect of this algorithm – constraint sampling – is discussed in detail in [27].

Theoretically, the value function can be obtained by solving the infinite dimensional LP (25.7). However, this is not feasible because there are an infinite number of decision variables and an infinite number of constraints. This technical difficulty can be addressed by constraining the function to the span of the basis functions. In particular, we consider solving

$$\min_{\mathbf{r}} \quad \int \sum_{k=0}^{K} r_k \phi_k(\mathbf{x}) \pi(d\mathbf{x}) \tag{25.9}$$

$$\text{s.t.} \quad L(\mathbf{x},\mathbf{u}) + F(\mathbf{x},\mathbf{u})^T \sum_{k=0}^{K} r_k \nabla \phi_k(\mathbf{x}) - \alpha \sum_{k=0}^{K} r_k \phi_k(\mathbf{x}) \leq 0 \quad \forall (\mathbf{x},\mathbf{u}) \text{ s.t. } S(\mathbf{x},\mathbf{u}) \leq \mathbf{0},$$

which is known as the approximate linear program (ALP). Notice that we have removed the technical condition $\sum_{k=0}^{K} r_k \phi_k(\cdot) \in C^2$ from the constraints of (25.9) since it always holds for the basis functions considered. However, there are still an infinite number of constraints in the ALP (25.9), so it cannot be solved exactly. We overcome this issue through constraint sampling; that is, only $\mathcal{M}$, a finite set of sampled states and control actions, is used to constrain variables. This results in what is known as the reduced linear program (RLP). Let $\mathcal{M} = \left\{(\mathbf{x}^{(1)},\mathbf{u}^{(1)}),\cdots,(\mathbf{x}^{(M)},\mathbf{u}^{(M)})\right\}$,

where M is the cardinality of $\mathcal{M}$. The RLP takes the form

$$\begin{aligned} \min_{\mathbf{r}} \quad & \frac{1}{M}\sum_{m=1}^{M}\sum_{k=0}^{K} r_k \phi_k(\mathbf{x}^{(m)}) \\ \text{s.t.} \quad & L(\mathbf{x}^{(m)},\mathbf{u}^{(m)}) + F(\mathbf{x}^{(m)},\mathbf{u}^{(m)})^T \sum_{k=0}^{K} r_k \nabla\phi_k(\mathbf{x}^{(m)}) - \alpha \sum_{k=0}^{K} r_k \phi_k(\mathbf{x}^{(m)}) \leq 0 \\ & \forall m = 1,2,\cdots,M \end{aligned} \tag{25.10}$$

The way in which constraints are sampled significantly impacts the performance of the resulting approximation, and our approach to constraint sampling is nontrivial. Roughly speaking, we sample the state/action pairs based on a randomized baseline strategy and then bootstrap this process. Interested readers can refer to the appendix of [27] for the constraint sampling algorithm. In practice, a rule of thumb is to choose the number of sampled states M to be 40 to 60 times the size of K, the number of basis functions. It has been observed in some studies [11] that RLP (25.10) might be in some sense over-constrained. We have also observed that the L_1 regularization of $\mathbf{r}$ can improve performance. These observations motivate the smoothed reduced linear program (SRLP) [11] with L_1 regularization:

$$\begin{aligned} \min_{\mathbf{r},\mathbf{s}} \quad & \frac{1}{M}\sum_{m=1}^{M}\sum_{k=0}^{K} r_k \phi_k(\mathbf{x}^{(m)}) + \epsilon||\mathbf{r}||_1 \\ \text{s.t.} \quad & L(\mathbf{x}^{(m)},\mathbf{u}^{(m)}) + F(\mathbf{x}^{(m)},\mathbf{u}^{(m)})^T \sum_{k=0}^{K} r_k \nabla\phi_k(\mathbf{x}^{(m)}) - \alpha \sum_{k=0}^{K} r_k \phi_k(\mathbf{x}^{(m)}) \leq s^{(m)} \\ & \forall m = 1,2,\cdots,M \\ & s^{(m)} \geq 0 \quad \forall m = 1,2,\cdots,M \quad \text{and} \quad \sum_{m=1}^{M} s^{(m)} \leq \theta \sum_{m=1}^{M} |L(\mathbf{x}^{(m)},\mathbf{u}^{(m)})|, \end{aligned} \tag{25.11}$$

where $\mathbf{s} = \left[s^{(1)}, s^{(2)}, \cdots, s^{(M)}\right]$. There are two parameters to be specified in (25.11), θ and ϵ. The variable $\theta \geq 0$ characterizes the extent to which the constraints of the RLP are relaxed, and $\epsilon \geq 0$ characterizes a penalty on the magnitude of $\mathbf{r}$, in the sense of the L_1 norm. We have observed that the selection of θ and ϵ influences $\mathbf{r}_{srlp}$, the solution of (25.11). In practice, θ and ϵ are either pre-specified, or chosen based on line search.

25.4.3 Solving Sub-Problems

Once the approximate value function $\tilde{J}$ is available, control actions $\tilde{\mathbf{u}}$ can be obtained by solving (25.5), with J^* replaced by $\tilde{J}$. A useful observation relevant to oil production problems can significantly simplify this step. Consider the reservoir dynamics $d\mathbf{x}/dt = F(\mathbf{x}, \mathbf{u})$ and constraints $S(\mathbf{x}, \mathbf{u}) \leq 0$. In most cases, although $F(\mathbf{x}, \mathbf{u})$ and $S(\mathbf{x}, \mathbf{u})$ are nonlinear functions of the state $\mathbf{x}$, F is affine in the control action $\mathbf{u}$ for fixed $\mathbf{x}$. In addition, for common constraints used in practice, such as maximum/minimum flow rates, S is also affine in the control action $\mathbf{u}$ for fixed $\mathbf{x}$. [2]

This means that

$$\begin{aligned} F(\mathbf{x}, \mathbf{u}) &= F_1(\mathbf{x}) + F_2(\mathbf{x})\mathbf{u} \\ S(\mathbf{x}, \mathbf{u}) &= S_1(\mathbf{x}) + S_2(\mathbf{x})\mathbf{u}, \end{aligned}$$

thus, the sub-problem at time t is

$$\begin{aligned} \max_{\mathbf{u}} \quad & L\left(\mathbf{x}(t), \mathbf{u}\right) + \left[\nabla \tilde{J}\left(\mathbf{x}(t)\right)\right]^T F_2(\mathbf{x}(t))\mathbf{u} \\ \text{s.t.} \quad & S_1(\mathbf{x}(t)) + S_2(\mathbf{x}(t))\mathbf{u} \leq \mathbf{0}. \end{aligned} \tag{25.12}$$

This problem (25.12) is easy to solve if the instantaneous payoff function L is concave in $\mathbf{u}$. For example, with the instantaneous payoff function L defined in (25.2), the sub-problem at each time reduces to a low-dimensional linear program.

25.4.4 Adaptive Basis Function Selection and Bootstrapping

The ADP algorithm described thus far is "open-loop" in the sense that once a new control $\tilde{\mathbf{u}}$ is available, it does not get used to further improve the result. In this subsection, we introduce two advanced techniques, adaptive basis function selection and bootstrapping, to "close the loop" and enhance the performance of the ADP algorithm.

Adaptive Basis Function Selection As discussed earlier, in practical implementations, we need to select a subset of basis functions from a large set of candidates. One

[2]For interested readers, this is the case because the so-called well transmissibility does not depend on BHP or flow rate.

way to effectively choose basis functions is through use of adaptive basis function selection. Specifically, we initialize the set of basis functions $\mathcal{F}$ as a set containing only the constant basis function $\phi_0(\cdot) = 1$ (or a known set of "good" basis functions such as those selected in the previous bootstrapping round). At each iteration, we choose a new basis function and add it to $\mathcal{F}$. Our rule in selecting the new basis function is that, compared to other candidate basis functions, including the new basis function in $\mathcal{F}$ results in the largest increase in the net present value. We continue to add basis functions until the addition of any new basis function leads to a decrease in net present value. In other words, adaptive basis function selection is a greedy algorithm for adding new basis functions. Refer to [27] for a detailed description of this algorithm.

At each step of the adaptive basis function selection procedure, the algorithm iterates over all of the candidate basis functions to choose the optimal basis function to add to $\mathcal{F}$. This approach can be time consuming. However, similar to the constraint sampling algorithm, parallel computation can significantly accelerate this algorithm.

Bootstrapping Another technique to further improve the performance of our ADP algorithm is bootstrapping. Specifically, to construct basis functions based on POD and sample the constraints for SRLP (25.11), we need to sample states and state/action pairs based on a known strategy μ. Theory [10] suggests that, to obtain the best results, we should use an optimal control policy when sampling. Since an optimal policy is not available (the objective of the ADP algorithm is to find a policy close to optimal), we sample using a baseline strategy μ_0, which inevitably leads to some performance loss.

Bootstrapping is proposed to address this issue. Specifically, we start from a baseline strategy μ_0, then apply the above-described ADP algorithm to compute an approximate value function $\tilde{J}_1$, which generates a new strategy μ_1. Then, we sample states and state/action pairs based on μ_1, and apply the ADP algorithm again to obtain a new approximate value function $\tilde{J}_2$ and policy μ_2. We repeat this procedure until it no longer increases net present value. A detailed description of the bootstrapping algorithm is provided in [27].

Figure 25.2 illustrates the architecture of our ADP algorithm with the incorporation of adaptive basis function selection and bootstrapping.

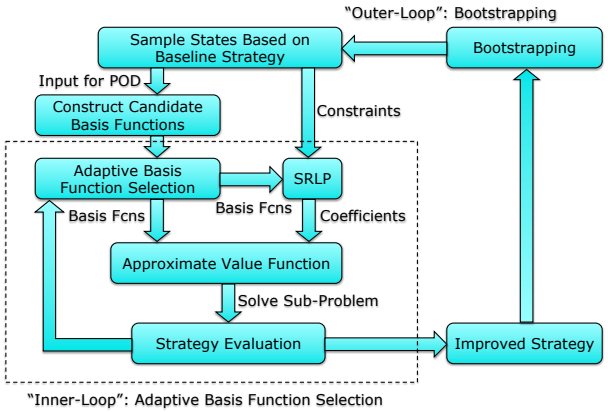


Figure 25.2: Architecture of the overall ADP algorithm.

25.4.5 Computational Requirements

We now briefly discuss the computational requirements of our proposed ADP algorithm. As is the case for other production optimization algorithms, most of the computation in ADP is consumed by the reservoir simulations. Thus it is appropriate to assess the computational demands for the proposed ADP algorithm in terms of the required number of simulations.

As shown in Figure 25.2, the ADP algorithm uses the reservoir simulator to sample states and evaluate the strategy $\tilde{\mu}$. As discussed above, in general, SRLP (25.11) requires $40K$ to $60K$ state samples, where K is the number of basis functions, and each state sample requires one simulation.

Adaptive basis function selection requires a large number of additional simulations. For this approach, assume there are $\tilde{K}$ candidate basis functions and we are allowed to use up to K_{max} basis functions. One obvious upper bound on the number of required simulations is $60K_{max} + \tilde{K}K_{max}$. However, this bound is far from tight for the following reasons: first, as is observed in Case 2 below, for large K_{max} the number of basis functions ultimately used (K) tends to be much less than K_{max}.

Second, the ADP algorithm does not need to reevaluate the strategy if the computed coefficient of the new basis function is zero, which happens frequently due to the L_1 regularization in SRLP (25.11). Thus the number of required simulations in practice is much less than the theoretical bound above.

For the ADP algorithm with bootstrapping, assuming the algorithm applies the bootstrapping B times, the required number of simulations will be B times the above results. In practice, it has been observed that the performance tends to degrade after several iterations of bootstrapping (see [14]; this is also observed in the examples here). Hence, $B \leq 5$ in most cases.

It is evident that ADP runtime scales with the number of (candidate) basis functions. In general, the use of more candidate basis functions in adaptive basis function selection will improve algorithm performance though it requires more computation. Thus, there is a trade-off between performance and computational effort. It would be of interest to further investigate this trade-off in future work. As we have discussed above, it is important to note that both state sampling and adaptive basis function selection can be implemented in parallel. Thus, by using multiple processors, the elapsed time for the ADP algorithm need not be excessive.

One way to design the candidate basis function pool is to consider only one-term polynomials. In this case, the number of candidate basis functions is $\tilde{K} = O(N_p + N_S + N_w)$, where N_w is the number of wells and N_p and N_S are determined during POD. However, in some cases, the "cross terms" (i.e., functions with more than one term) are useful in approximating the value function. Of course, adding all such cross terms will result in an intractably large candidate basis function pool. One way to address this issue is to add only the cross terms between the first few columns of $\mathbf{\Phi}$ (which correspond to the largest singular values) to the candidate basis function pool.

25.5 SIMULATION RESULTS

In this section, we apply the above proposed ADP algorithm to two example cases and compare our results against various baseline strategies. The first example is

a single-phase primary production case, and the second example is a two-phase oil-water case with general (nonlinear) constraints.[3] Both examples involve three-dimensional reservoir models. In our implementation we use Stanford's General Purpose Research Simulator, GPRS [6, 17] for all reservoir simulations and CPLEX, a commercial LP solver, to solve the SRLP (25.11). Other components of the ADP algorithm are implemented in MATLAB.

We have also applied the proposed ADP algorithm to three different examples involving two-dimensional reservoir models in [27]. In that work we compared the performance of the ADP algorithm with that of the gradient-based method described in [21]. Interested readers can refer to [27] for the detailed results.

Case 1: Primary Production with Penalty Terms

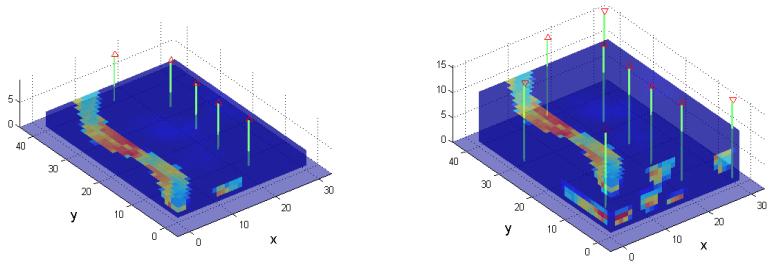
In this first example, we apply the proposed ADP algorithm to a single-phase primary production case. In this example the reservoir contains only oil and no fluid is injected into the reservoir. Thus the state vector $\mathbf{x}$ only includes the pressure of each grid block. The reservoir dynamics are described by a linear flow equation in this case. The geological model is the top three layers of the Stanford VI synthetic reservoir model [8]. The model contains $30 \times 40 \times 3$ grid blocks and has five production wells, each of which is controlled by a bounded BHP. The geological model is shown in Figure 25.3(a) (the x-direction permeability field is the quantity depicted). We simulate for 300 days with 300 control periods. The instantaneous profit function is

$$L(\mathbf{x}, \mathbf{u}) = \text{revenue from producing oil} + \lambda \sum_{i=1}^{5} \log(\mathbf{u}_i - LB_i),$$

where $\mathbf{u}_i$ is the BHP for producer i and LB_i is its lower bound. The logarithm term penalizes BHPs that are close to the lower bound. Including this term (in this form) enables the analytical determination of the global optimum (i.e., globally optimal control and the associated NPV of the cumulative profit) using DP. Key parameters for Case 1 are summarized in Table 25.1. Note that STB stands for stock tank barrel

[3] Mathematically, these constraints are nonlinear inequalities involving $\mathbf{x}$ and $\mathbf{u}$. However, as discussed in Subsection 25.4.3, for a given state $\mathbf{x}$, they are affine inequalities of $\mathbf{u}$.

(1 STB = 0.1590 m^3) and 14.50 psi = 1 bar (STB and psi are standard units in petroleum engineering).



(a) Case 1 (b) Case 2

Figure 25.3: Geological models (x-direction permeability, where blue indicates low and red indicates high permeability) and well configurations for simulation examples. Production wells are indicated by green lines with upward-pointing triangles and injection wells by green lines with downward-pointing triangles.

Table 25.1: Parameters for Case 1

BHP range for Producer 1	800-5000 psi	BHP range for Producer 2	800-5000 psi
BHP range for Producer 3	1200-5000 psi	BHP range for Producer 4	1000-5000 psi
BHP range for Producer 5	500-5000 psi	Oil price	\$40/STB
Logarithm penalty coefficient λ	1×10^4	Discount rate α	1×10^{-2}

We apply the ADP algorithm with both adaptive basis function selection and bootstrapping. Note in this case the projection matrix from POD is $\mathbf{\Phi} = U_p(1 : N_p)$,

and we choose $N_p = 12$. The candidate basis function pool includes (1) a constant basis function $\phi_0 = 1$, (2) average pressure $\bar{p}$ of the entire reservoir, (3) average pressures in near-producer regions, and (4) all the first-order polynomials defined on span($\mathbf{\Phi}$). There are a total of $\tilde{K} = 19$ candidate basis functions, and we set $K_{max} = \tilde{K} = 19$.

The progress of the ADP algorithm is presented in Figure 25.4. We start

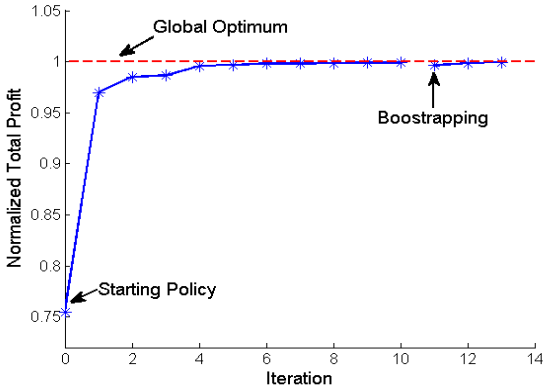


Figure 25.4: Performance of ADP for Case 1, where the normalization is with respect to the global optimum.

from an (arbitrary) constant control strategy $\mu_0(\mathbf{x}) = LB + 10\mathbf{e}$, where $LB = [LB_1, \cdots, LB_5]^T \in \Re^5$ is the vector encoding the lower bounds of the producer BHPs and $\mathbf{e}$ is the vector of all ones. With this starting strategy, POD characterizes a subspace of dimension 12 that captures more than 99.9999% of the variation in pressure (in the sense of singular values). For the constraint sampling, we use randomized μ_0 to sample 800 states, with randomization magnitude $\eta = 100$ psi. Then we apply the adaptive basis function selection algorithm, with the initial basis function set $\mathcal{F} = \{\phi_0(\cdot) = 1\}$. In order to compute the coefficients based on SRLP (25.11), we choose $\epsilon = 0.1$ and choose θ based on a line search from 0 to 0.2. As is evident in Figure 25.4, after adaptively adding 10 basis functions, addition of any new basis functions in $\mathcal{F}$ leads to a decrease in NPV. Thus, the ADP algorithm terminates

adaptive basis function selection after the 11th iteration and records the associated sub-optimal control strategy μ_1. The result from μ_1 is 99.88% of the global optimum.

The ADP algorithm then bootstraps once using control strategy μ_1, that is, it reconstructs $\mathbf{\Phi}$ based on μ_1 and then performs the constraint sampling based on randomized μ_1, followed by another round of adaptive basis function selection. As in the first round, we use $\eta = 100$ psi, $\epsilon = 0.1$ and choose θ based on a line search from 0 to 0.2. The difference here is that the adaptive basis function selection starts from the 11 basis functions selected in the first bootstrapping round. We observe that, after adding two new basis functions, neither adding a new basis function nor bootstrapping further increases NPV. Thus, the ADP algorithm terminates, with a result that is 99.93% of the global optimum. We note that the ADP algorithm was also applied with different starting policies and achieved similar results.

In addition to the global optimum, another interesting control strategy is the myopic policy (the policy that aims at maximizing the instantaneous profit at each time). In this case, the myopic policy achieves 93.51% of the global optimum. Thus we see that the ADP result is not only very close to the global optimum for this case, it is also significantly better than the result using the myopic policy.

Although this case is quite simple, it represents one of the few cases where the global optimum can be obtained analytically. It is significant that, for a case where the global optimum is known, ADP does indeed provide a result that is very close to the global optimum.

Case 2: Waterflood with General Constraints

We now consider a two-phase (oil-water) model with general constraints. The geological model, which contains $30 \times 40 \times 10$ grid blocks, is again a portion of the Stanford VI model. Flow is driven by four injection wells and five production wells, as indicated in Figure 25.3(b). We impose the following constraints at each time t: (1) maximum field water injection rate, (2) minimum field oil production rate, (3) maximum field liquid (water plus oil) production rate, (4) maximum watercut (fraction of water in the produced fluid) for each production well, and (5) upper and lower bounds on BHP of each well.

In this case, the reservoir dynamics are described by nonlinear equations, and most of the constraints listed above are also nonlinear. The instantaneous profit function is specified by (25.2). We simulate for 365 days with 10 control periods. The constraints and parameters are given in Table 25.2.

Table 25.2: Parameters for Case 2

BHP range for producers	2500-6000 psi	BHP range for injectors	4500-9000 psi
Maximum field water injection rate	4×10^4 STB/day	Minimum field oil production rate	1×10^4 STB/day
Maximum field liquid production rate	5×10^4 STB/day	Maximum watercut	0.7
Oil price	\$50/STB	Water production cost	\$10/STB
Water injection cost	\$5/STB	Discount rate α	1×10^{-2}

For oil-water problems with the constraints specified above, it can be difficult to find a feasible control strategy without applying an optimization technique. In particular, the myopic policy is infeasible for this case. However, for the subproblems (25.12), feasible control can be found as follows. We randomly sample parameters $\mathbf{g} \in \Re^9$ (each component of $\mathbf{g}$ is sampled i.i.d. from a uniform distribution over $[-1, 1]$) and replace the unknown affine function $L(\mathbf{x}(t), \mathbf{u}) + [\nabla J^*(\mathbf{x}(t))]^T F_2(\mathbf{x}(t))\mathbf{u}$ by $\mathbf{g}^T\mathbf{u}$. At time t with state $\mathbf{x}(t)$, we then solve the following LP to obtain a feasible control at time t:

$$\begin{aligned} \max_{\mathbf{u}} \quad & \mathbf{g}^T\mathbf{u} \\ \text{s.t.} \quad & S_1(\mathbf{x}(t)) + S_2(\mathbf{x}(t))\mathbf{u} \leq \mathbf{0}. \end{aligned} \tag{25.13}$$

If (25.13) is infeasible at time t, we restart this procedure from time 0. We use this approach to generate 100 feasible control strategies, and choose the best one as the baseline for Case 2.

Both adaptive basis function selection and bootstrapping are used for this case. We start the ADP algorithm by using the above-derived baseline strategy to construct the candidate basis function pool through POD. Specifically, we partition the projection matrix as $\mathbf{\Phi} = [\mathbf{\Phi}_1, \mathbf{\Phi}_2]$, where $\mathbf{\Phi}_1$ includes the first six left singular vectors. The candidate basis function pool includes (1) a constant basis function $\phi_0 = 1$, (2) one-term polynomials of average reservoir pressure and saturation and oil-water ratio of the entire reservoir up to third order, (3) one-term polynomials of the average oil saturations in near-producer regions up to third order, (4) all the polynomials defined on span($\mathbf{\Phi}_1$) up to third order, and (5) all the other one-term polynomials defined on span($\mathbf{\Phi}$) up to third order. There are thus a total of $\tilde{K} = 340$ candidate basis functions and we choose $K_{max} = 50$. Then we use the randomized baseline strategy (with the randomization magnitude $\eta = 100$ psi) to sample $M = 2000$ constraints for the SRLP (25.11). Finally, we start adaptive basis function selection with the initial basis function set $\mathcal{F} = \{\phi_0(\cdot) = 1\}$. For a given choice of basis functions, the coefficients are computed by SRLP (25.11), and we set $\epsilon = 5 \times 10^{-2}$ and choose θ by a line search from 0 to 0.05. We observe that, after adding seven basis functions, inclusion of any new basis functions decreases the NPV of the cumulative profit. Compared with the baseline, without any bootstrapping, the ADP algorithm achieves a 16.03% improvement.

The ADP algorithm then bootstraps once using the obtained strategy. We re-construct the candidate basis function pool and re-sample the constraints. Then we start the basis function selection from the eight basis functions chosen in the first bootstrapping round. We observe that, after adding three new basis functions, NPV is not increased further by adding new basis functions or bootstrapping. Thus the ADP algorithm terminates, at which point it has achieved an 18.67% improvement compared with the baseline strategy. Optimization results for Case 2 are plotted in Figure 25.5.

Figure 25.6 presents simulation results together with the associated constraints over time for the near-optimal control obtained by the ADP algorithm. It is evident that the maximum water injection rate and maximum liquid production rate are tight at the beginning of the simulation, while the minimum oil production rate and watercut

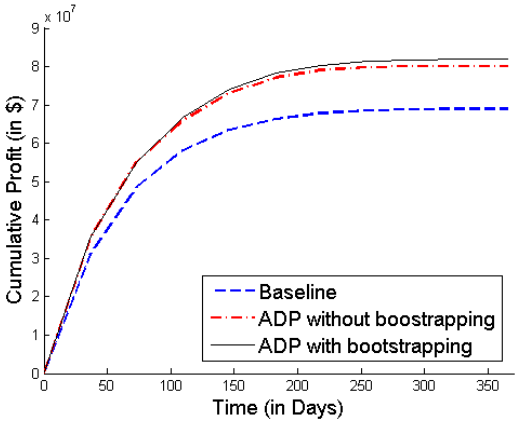


Figure 25.5: Comparison of ADP (both with and without bootstrapping) with baseline strategy for Case 2.

of Producer 4 are tight at the end of the simulation. In addition, for most of the wells, bounds on the BHPs are tight in all or some simulation time steps. The constraints that are non-tight over the entire simulation could be relaxed. In any event, this example demonstrates that our ADP algorithm does satisfy all of the constraints throughout the simulation.

25.6 CONCLUDING REMARKS

In this chapter, we developed an LP-based ADP algorithm for oil production optimization. We discussed the general algorithm, described how to efficiently construct and select basis functions, and how to compute their coefficients based on SRLP. Results were presented for primary production and water injection problems. Compared with the baseline strategies, the ADP algorithm was shown to perform well.

It is worth noting that the ADP algorithm proposed in this chapter should also be applicable for problems other than oil production. In fact, except for constructing some pre-specified candidate basis functions (such as average oil saturation) and the separation of pressure and saturation in POD, the other steps in the algorithm are not specific to the oil production problem. The main characteristics of oil production

problems that we exploited in our new ADP algorithm are (1) the availability of a reservoir simulator, which enables model-order reduction and control strategy evaluation, and (2) the tractability of the sub-problem (25.5) (since it is convex). These two characteristics together ensure an efficient evaluation of the approximate value function $\tilde{J}$, and hence render the adaptive basis function selection and bootstrapping computationally tractable. For optimal control problems in other fields that share these characteristics, we expect that some of the techniques proposed in this chapter, such as basis function construction using model-order reduction and the incorporation of adaptive basis function selection and bootstrapping, or even this ADP algorithm in its entirety, could be applied after suitable modifications.

Acknowledgments

We are grateful to the industry sponsors of the Stanford Smart Fields Consortium for partial funding of this work. We also thank David Echeverría Ciaurri and Huanquan Pan for useful discussions and suggestions.

REFERENCES

1. V. Artus, L. J. Durlofsky, J. Onwunalu, and K. Aziz. Optimization of nonconventional wells under uncertainty using statistical proxies. *Computational Geosciences*, 10:389–404, 2006.

2. K. Aziz and A. Settari. *Petroleum Reservoir Simulation*. Applied Science Publishers, 1979.

3. D. P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 1. Athena Scientific, 1995.

4. D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

5. D. R. Brouwer and J. D. Jansen. Dynamic optimization of waterflooding with smart wells using optimal control theory. *SPE Journal*, 9(4):391–402, December 2004.

6. H. Cao. *Development of Techniques for General Purpose Simulators*. PhD thesis, Stanford University, 2002.

7. M. A. Cardoso and L. J. Durlofsky. Use of reduced-order modeling procedures for production optimization. *SPE Journal*, 15(2):426–435, June 2010.

8. S. Castro. *A Probabilistic Approach to Jointly Integrate 3D/4D Seismic, Production Data and Geological Information for Building Reservoir Models*. PhD thesis, Stanford University, 2007.

9. A. S. Cullick, D. Heath, K. Narayanan, J. April, and J. Kelly. Optimizing multiple-field scheduling and production strategy with reduced risk. SPE paper 84239 presented at the 2003 SPE Annual Technical Conference and Exhibition, Denver, Colorado.

10. D. P. de Farias and B. Van Roy. On constraint sampling in linear programming approach to approximate dynamic programming. *Mathematics of Operations Research*, 29(3):462–478, August 2004.

11. V. V. Desai, V. F. Farias, and C. C. Moallemi. The smoothed approximate linear program. In *Advances in Neural Information Processing Systems 22*. MIT Press, 2009.

12. D. Echeverría Ciaurri, O.J. Isebor, and L.J. Durlofsky. Application of derivative-free methodologies for generally constrained oil production optimization problems. *International Journal of Mathematical Modelling and Numerical Optimization*, 2:134–161, 2011.

13. V. F. Farias and B. Van Roy. An approximate dynamic programming approach to network revenue management. `www.stanford.edu/~bvr/psfiles/adp-rm.pdf`. Working paper.

14. V. F. Farias and B. Van Roy. Tetris: A study of randomized constraint sampling. In G. Calafiore and F. Dabbene, editors, *Probabilistic and Randomized Methods for Design Under Uncertainty*. Springer-Verlag, 2006.

15. J. Han and B. Van Roy. Control of diffusions via linear programming. In G. Infanger, editor, *Stochastic Programming: The State of the Art, in Honor of George B. Danzig*, pages 329–354. Springer, 2011.

16. T. J. Harding, N. J. Radcliffe, and P. R. King. Optimization of production strategies using stochastic search methods. SPE paper 35518 presented at the 1996 European 3-D Reservoir Modeling Conference, Stavanger, Norway.

17. Y. Jiang. *Techniques for Modeling Complex Reservoirs and Advanced Wells*. PhD thesis, Stanford University, 2007.

18. P. W. Keller, S. Mannor, and D. Precup. Automatic basis function construction for approximate dynamic programming and reinforcement learning. In *Proceedings of the 23rd International Conference on Machine Learning*, 2006.

19. F. A. Longstaff and E. S. Schwartz. Valuing American options by simulation: a simple least-square approach. *Review of Financial Studies*, 14(1):113–147, 2001.

20. W. B. Powell. *Approximate Dynamic Programming*. John Wiley and Sons, 2007.

21. P. Sarma, L. J. Durlofsky, K. Aziz, and W. H. Chen. Efficient real-time reservoir management using adjoint-based optimal control and model updating. *Computational Geosciences*, 10:3–36, 2006.

22. H. P. Simão, A. George, W. B. Powell, T. Gifford, J. Nienow, and J. Day. Approximate dynamic programming captures fleet operations for Schneider National. *Interfaces*, 40(5):342–352, 2010.

23. G. J. Tesauro. Temporal difference learning and TD–Gammon. *Communications of the ACM*, 38:58–68, 1995.

24. J. N. Tsitsiklis and B. Van Roy. Optimal stopping of Markov processes: Hilbert space theory, approximation algorithms, and an application to pricing high-dimensional financial derivatives. *IEEE Transactions on Automatic Control*, 44(10):1840–1851, 1999.

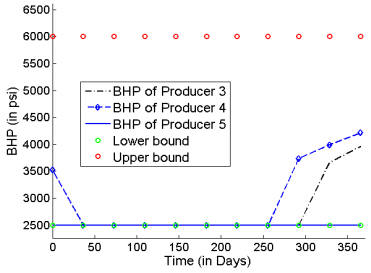
25. J. N. Tsitsiklis and B. Van Roy. Regression methods for pricing complex American-style options. *IEEE Transactions on Neural Networks*, 12(4):694–703, 2001.

26. B. Van Roy. Neuro-dynamic programming: overview and recent trends. In E. Feinberg and A. Shwartz, editors, *Handbook of Markov Decision Processes: Methods and Applications*. Kluwer, 2001.

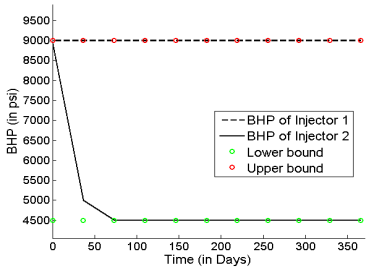
27. Z. Wen, L. J. Durlofsky, B. Van Roy, and K. Aziz. Use of approximate dynamic programming for production optimization. SPE paper 141677 presented at the 2011 SPE Reservoir Simulation Symposium, The Woodlands, Texas.

28. D. Zhang and D. Adelman. Dynamic bid-prices in revenue management. *Operations Research*, 55(4):647–661, 2007.

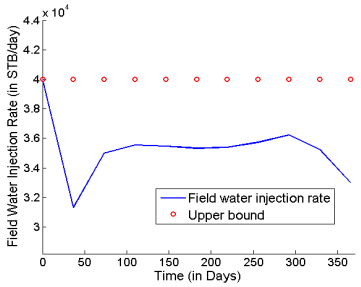
29. D. Zhang and D. Adelman. An approximate dynamic programming approach to network revenue management with customer choice. *Transportation Science*, 43:381–394, 2009.



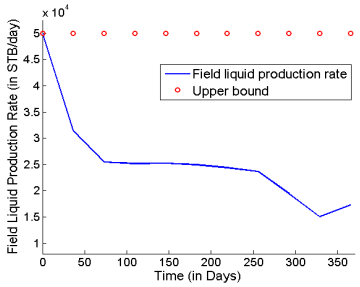
(a) Producer BHPs from ADP



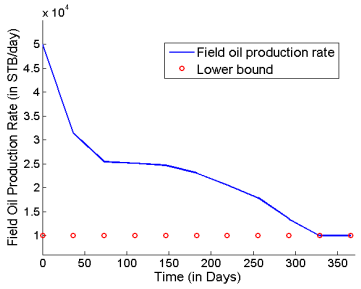
(b) Injector BHPs from ADP



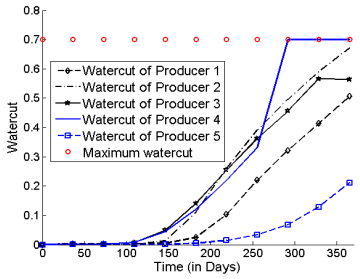
(c) Field water injection rate from ADP



(d) Field liquid production rate from ADP



(e) Field oil production rate from ADP



(f) Watercuts from ADP

Figure 25.6: Simulation results for the near-optimal control achieved by ADP for Case 2.