Machine Learning Strategies for Steam Injection Optimization in SAGD

by

Jose Leonardo Guevara Urdaneta

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Petroleum Engineering

Department of Civil and Environmental Engineering

University of Alberta

Abstract

Finding optimal steam injection policies in the context of Steam Assisted Gravity Drainage (SAGD) represents a major challenge due to the complex dynamics of the process. This complexity is reflected by: i) several concurrent sub-processes, *e.g.* heat transfer, counter-current flow, imbibition, ii) potential reservoir heterogeneity, and iii) the lagged nature of the process. As a result, conventional steam injection strategies or policies in SAGD are typically not a result of a formal optimization process but rather empirically found. Furthermore, available optimization methods exhibit important drawbacks such as, requiring the full mathematical description of the process (adjoint-optimization) or may not be suitable for long-term optimization (Model Predictive Control).

In this work, we propose two (2) alternatives to solve the cited challenge. The first alternative is the use of reinforcement learning (RL), in which no information of the physical SAGD phenomena is needed and can potentially optimize for long-term cumulative performance. In particular, we present the implementation of the two (2) main RL approaches: action-value function and policy gradient, for one and multiwell applications, respectively. In both implementations, obtained optimal steam injection policies exhibit a significant improvement both with respect to the initial (random) policies and constant steam injection strategies. Furthermore, these optimal policies exhibit two distinctive regions: initially, an increase or slight increase of steam injection rates (Region 1), and afterwards, a sharp decrease until reaching the minimum value (Region 2). This shape suggests that for optimal SAGD operations: i) steam chamber expansion is key until the overburden is reached (Region 1), afterwards, reservoir temperature should be kept high and ii) pressure plays a vital role until the steam chamber reaches the overburden, afterwards temperature is the driving mechanism of oil production.

Reinforcement learning although represents a promising solution to the SAGD optimization challenge, it may require the continuous execution of a potentially computationally expensive numerical reservoir simulation model. As a result, the second alternative presented in this work is the use of dynamic surrogate modeling and optimization (DSMO) framework. In particular, we propose a methodology to build surrogate models that can provide fast approximations of time-varying outputs (*e.g.* daily oil production rates) of the SAGD process which could be used to solve the cited optimization problem.

The proposed method represents an improvement of the conventional recursive based prediction approach in which a one-step prediction model(s) is identified and then used recursively to predict n-steps in the future. We propose the use of a second model that can capture the variability of the residual exhibited by the first model, and then act as a correction term. The underlying assumption, which is empirically discussed, is that the residuals of the recursive approach are correlated with time and thus can be generalized over the input space. The methodology consists of an extension of the traditional surrogate model and optimization (SMO) framework that considers non time-varying variables and consists of identifying surrogate models of any physical process using samples from high fidelity models.

We test the proposed approach on a multi well reservoir simulation model using recurrent neural networks for both the one-step prediction model and the residual or correction model. Results show that the proposed approach offers significantly better prediction capabilities as compared to the conventional recursive approach. In particular, the corrected model is able to capture output variability (R2) and reduce error (Mean Absolute Percentage Error) consistently over several statistical realizations of the selected samples. Furthermore, we can show these results are similar when considering different limited and extended samples sizes, suggesting the efficiency of the method.

Preface

The research presented in this dissertation was carried out under the supervision of Dr. Japan Trivedi. In particular, his contribution was on the concept formulation of the research and, review and edition of the initial draft of this document.

A version of *Chapter 3* has been published as Guevara, JL, Patel, RG & Trivedi, J., 2021 "Optimization of steam injection in SAGD using reinforcement learning", *Journal of Petroleum Science and Engineering*, 206, 108735. Another preliminary version has been published in the 2018 SPE International Heavy Oil Conference and Exhibition held in Kuwait City, Kuwait. My specific contributions was in the problem formulation, design, and implementation of the reinforcement learning algorithms and results analysis. Dr. Rajan Patel developed the reservoir model used as case study.

A version of *Chapter 4* will be presented in the 2023 SPE Latin American and Caribbean Petroleum Engineering Conference to be held in Port of Spain, 14-15 June, Trinidad and Tobago as Guevara, JL & Trivedi, J. "Reinforcement learning for multi-well SAGD optimization: a policy gradient approach" and a journal version will be submitted to *Journal of Petroleum Science and Engineering.* I was responsible for the problem formulation, design, and implementation of the reinforcement learning algorithms and results analysis. Dr. Rajan Patel developed the reservoir model used as case study.

A version of *Chapter 5* is in progress for submission as Guevara, JL & Trivedi,

J. 2023 "Towards a machine learning based dynamic surrogate modeling and optimization of steam injection policy in SAGD" in the *Journal of Petroleum Science and Engineering*. Also, a preliminary version was presented in the 2022 SPE Western Regional Meeting held in Bakersfield, California, USA. I designed and coded the proposed algorithm and investigated the results using statistical measures. Mr. Najmudeen Sibaweihi developed the reservoir model used as case study.

A version of *Chapter 6* will be presented in the 2023 SPE Canadian Energy Technology Conference and Exhibition to be held in Calgary, Alberta, Canada 15-16 March, as Guevara, JL & Trivedi, J. "Dynamic surrogate model for prediction of oil production rates in SAGD processes". Additionally, a more complete version will be submitted to *Journal of Petroleum Science and Engineering*. In both cases, I contributed to the development and implementation of the dynamic surrogate model methodology, results and statistical analysis. Mr. Najmudeen Sibaweihi developed the reservoir model used as case study.

If knowledge isn't a right, then it must be a left

– Silvio Rodriguez.

Acknowledgements

I would like to express my appreciation to Dr. Japan Trivedi for his support in the completion of this work...

To my family for their constant words of encouragement...

And to all my friends all around the world...

Additionally, ss a part of the University of Alberta's Future Energy Systems (FEs) research initiative, this research was made possible in part thanks to funding from the Canada First Research Excellence Fund (CFREF). The authors also wish to acknowledge financial support from the University of Alberta through Doctoral Recruitment Scholarship and Computer Modelling Group (CMG) for providing academic licenses for the thermal reservoir simulator STARS and CMOST.

Contents

List of Tables

List of Figures

Nomenclature

α	Learning rate
$\bar{y}$	Average of the true values of the variable of interest
β	Regularization control parameter
β_p	Vector of parameters of time series models
ΔE	Objective function deterioration in Simulated Annealing
Δ	Constant value by which steam injection rate values can be modified
ϵ	Residuals
ϵ_g	Probability of taking a random action
ϵ_{decay}	Decay factor of ϵ_g
γ	Discount factor
$\hat{\mathbf{g}}$	Surrogate function that describes the dynamic of the system
$\hat{q}$	Approximated action-value function
$\hat{ym}$	Plant output
$\hat{y}$	Predicted values of the variable of interest
$\hat{z}$	Normalized prediction variable of interest
λ	Lagrange multipliers
μ	Mean of random process
ϕ	Parameters of the exogeneous input component
π	Policy
Ψ	Parameters of the general time series model

ψ	Parameters of the moving average component
σ	Standard deviation
τ	Parameters of the autoregressive component
$\mathbf{g}$	Reservoir dynamics
$\mathbf{p}$	Descent direction in BFGS
$\mathbf{u}$	Vector containing the steam injection rates at every time step
$\mathbf{u}_k$	Steam injection rate at iteration k
$\mathbf{u}_{max}$	Upper bounds of the steam injection rates at every time step
$\mathbf{u}_{min}$	Lower bounds of the steam injection rates at every time step
$\mathbf{w}$	Parameters of the approximated action-value function
$\mathbf{W}_1$	Set-point weight matrix
$\mathbf{W}_2$	Action control weight matrix
θ	Parameters of the policy
$\tilde{c}$	Output of the second part of the input gate
$\tilde{ym}$	Set-point
A	Specific action
a	Generic action
A_p	Number of individuals or particles in Particle Swarm Optimization
B	Hessian approximation
b	Exogeneous input order of the model
b_c	Bias of the second part of the input gate
b_f	Bias of the forget gate
b_i	Bias of the first part of the input gate
b_o	Bias of the output gate
c	Update version of the cell state
C_{steam}	Cost of steam generation [USD/STB]

C_{water}	Cost of produced water handling [USD/STB]
C_{water}	Cost of water production [USD/STB]
E	Loss function for SGR
e	Sequence of white noise
f	Output of the forget gate
G	Expected return
G_d	Process model
H	Hamiltonian function
h	Hidden state
H_d	Disturbance model
i	Annual discount factor [fraction]
J	Performance measure or objective function
k	Iteration number
m	Number of actions
N	Moving optimization horizon
n	Number of considered well pairs
n_c	Number of considered hydrocarbon components
n_{gb}	Number of gridblocks
Np	Cumulative oil production
o	Output of the LSTM
p	Autoregressive order of the model
P_o	Oil price [USD/STB]
q	Action-value function
q_d	Moving average order of the model
q_o	Oil production rate [STB/DAY]
q_s	Steam injection rate [STB/DAY]

q_w	Water production rate [STB/DAY]
$q_{o,j}$	Oil production rate of the j-th well [STB/day]
$q_{s,j}$	Steam injection rate of the j-th well [STB/day]
$q_{w,j}$	Water production rate of the j-th well
$R2$	Coefficient of determination
R	Return
R_{eg}	Regularization term
S	Specific state
s	Generic state
T	Production horizon
t	Time step
t_{drop}	Time in which steam injection rates decreases drastically
t_{ref}	Time reference for NPV calculations
u	Control action, design variable or steam injection rate
W_c	Weights of the second part of the input gate
W_f	Weights of the forget gate
W_i	Weights of the first part of the input gate
W_o	Weights of the output gate
Wi	Cumulative steam injection
Wp	Cumulative water production
x	State of the system
y	Current output value
y	True values of the variable of interest
Z	Prediction horizon
z	Prediction variable of interest

Chapter 1

Introduction

1.1 Background

A steam-assisted gravity drainage (SAGD) oil recovery process [10] consists of drilling two horizontal wells, an upper steam injection well and a lower production well placed a few meters vertically apart. Steam is then injected through the top well, it will ascend throughout the reservoir due to density difference and heat is transferred to the surrounding oil by conduction. The part of the reservoir impacted by the steam is referred to as the steam chamber (Figure 1.1) and as the injection continues it will expand vertically and horizontally. The heat transfer causes the viscosity of the oil to be significantly reduced (by several orders of magnitude) the steam condensates, and as a consequence, the condensate and mobilized oil will descend to the producer well driven by gravity forces and eventually reaches the surface.

In this context, finding the steam injection rate at every time step or policy, that maximizes cumulative performance (*e.g.* net present value) over the entire production horizon represents a major challenge due to the complexity of the phenomena. This complexity is expressed in part by the number of sub-processes that occur at the same time, such as, heat transfer [22], [30], counter-current flow [17], [69], co-current flow, water imbibition, emulsification [17] and steam fingering (Figure 1.2). All these

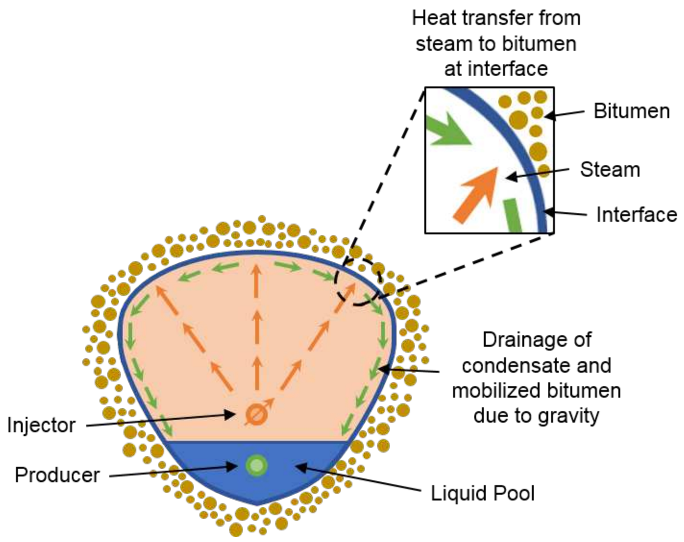

Heat transfer from steam to bitumen at interface
Bitumen
Steam
Interface
Drainage of condensate and mobilized bitumen due to gravity
Injector
Producer
Liquid Pool

Figure 1.1: Cross-section of the steam chamber developed in SAGD. Inset image shows heat transfer between steam and bitumen at the interface

sub-processes depend directly on the operating conditions (*e.g.* steam injection rate) and affect the performance of the process.

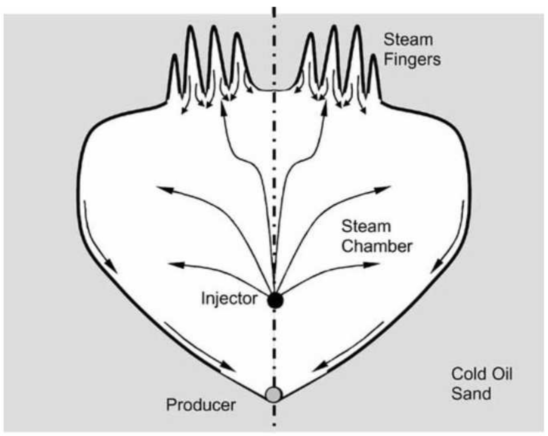


Figure 1.2: Exaggerated view of steam fingers during steam chamber expansion [36]

Another significant contribution to the complexity of the SAGD process is the presence of reservoir heterogeneity [51], [52], [90]. Heterogeneity is manifested in alternate pairs of sand and shale rock throughout the reservoir. These shale rocks act as barriers that prevent fluid flow and result in non-uniform growth of the steam chamber across the reservoir and may limit drainage path (Figure 1.3). Furthermore, shale rocks typically exhibit high water saturation, thus significant heat loss is expected as the steam chamber comes into contact with the shale rock due to the water's high heat capacity.

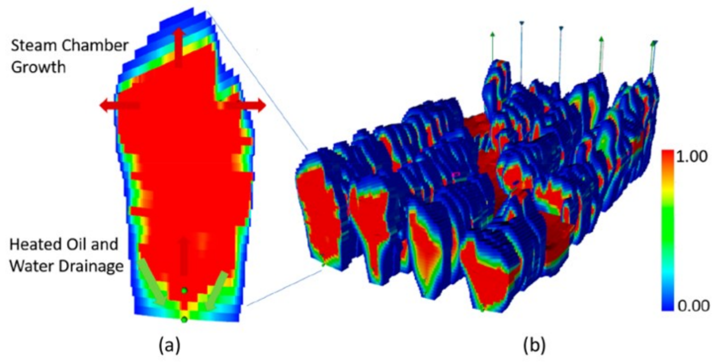


Figure 1.3: Non-uniform growth of the steam chamber due to reservoir heterogeneity [111]

Moreover, SAGD is a process that is significantly lagged, *i.e.* the effect of an injection rate applied at time t may be significantly delayed. Additionally, in some cases it may be better to sacrifice immediate production to maximize cumulative performance. For instance, a common start-up SAGD strategy is *pre heating* [29], [37], which consists of establishing thermal communication between the wells by injecting steam through both wells (injector and producer) for several months. This strategy has no immediate benefit but has been proven to significantly increase the long-term efficiency of the SAGD process.

The problem of finding the optimal steam injection policy further escalates considering that most industrial-scale applications of SAGD consist of dozens of well pair configurations operated simultaneously. In this configuration, there is typically a single steam source for all the well pairs, so the injection strategy is restricted by its maximum steam generation capacity. So, the problem is now to find the optimal steam injection policy for each well pair subject to the available steam generation, this is referred to as the steam allocation problem [41], [66], [110].

In field applications, steam injection policies used in SAGD processes although may fluctuate in time, are typically not the result of a formal optimization process. It is rather an empirical process that combines factors such as, reservoir characteristics, completions design for the wells, fluid dynamics within the wellbore, etc.

1.2 Problem Statement

Optimization problem In this work, we are interested in formulating the problem of finding optimal steam injection policies as an optimization. Mathematically this can be expressed as:

For a given SAGD configuration with n injector and producer wells (pairs) and a given production horizon with T time steps (*e.g.* daily), the objective is to find the steam injection policy that maximizes a particular performance measure (*e.g.* net present value) while accounting for the complex reservoir dynamics. Formally, the optimization problem can be written as:

$$\begin{aligned} \max_{\mathbf{u} \in \mathbb{R}^{n \times T}} \quad & J = \sum_{t=1}^{T} J_t(x_t, u_t) \\ \text{s.t.} \quad & \mathbf{u}_{min} \geq \mathbf{u} \geq \mathbf{u}_{max} \\ & x_{t+1} = \mathbf{g}(x_t, u_t) \end{aligned} \tag{1.1}$$

where J is a scalar performance measure, $u_t \in \mathbb{R}^{n \times 1}$ is the steam injection rates for all n wells at time t, x_t is the state matrix of the SAGD dynamic system (*e.g.* pressure and saturation distribution) at time t, $\mathbf{u}_{min}$ and $\mathbf{u}_{max}$[1] are the upper and lower bounds of the steam injection rate, and g is a function that represents the SAGD dynamic system

The optimization problem expressed by Equation 2.1 can be solved using two (2) main approaches: static and static optimization. Static optimization makes reference to settings in which the design or control variables are time-independent. Thus, each member of the vector $\mathbf{u}$ is treated as independent from each other and independent in time. In this case, conventional gradient or non-gradient-based methods, such as conjugated gradients, biologically inspired algorithms [6], [23], [31], [68], etc. Although this approach is easy to implement, as the time step size decreases or the production horizon T increases, the number of variables could make the problem computationally prohibited.

The second approach considers Equation 2.1 as an optimal control or trajectory optimization problem in which the time-dependency of the variables is considered. This problem (Equation 2.1) has typically been solved using adjoint-based optimization [48], model predictive control - MPC [85] and, more recently proposed, reinforcement learning [38], [45], [58], [97]. In this context, Reinforcement learning (RL) has the potential to overcome the shortcomings of the other optimal control strategies in that, it doesn't require a full mathematical description of the phenomena (adjoint-optimization) and can potentially optimize for long-term cumulative performance or reward (MPC).

In general, in RL, an agent is trained specifically to behave optimally to maximize the cumulative performance of a given process with no previous knowledge of the process [99]. The behavior of the agent is represented by actions taken (*e.g.* increase

[1]To clarify, the bold and non-italic representation of $\mathbf{u}$ express that contains all the steam injection rates for each well at each time step.

steam injection rate) at every time step; as a result, these actions will impact the process or environment and change its state and offer the agent a scalar reward (e.g., net present value). The ultimate goal of the agent is to maximize cumulative reward.

One important by-product of RL is that once found, the optimal policy may allow us to have a better understanding of the phenomenon or "how the world works". So for example, one of the most popular implementations (AlphaGo/AlphaGo Zero) of RL is in playing the abstract-strategy game Go [94]. Alpha Go not only defeated the world's top players but its style of play has been characterized by odd-looking, since -as opposed to what human players tend to do- sometimes it plays moves that lose material because it is seeking to maximise its probabilities of winning rather than maximise territorial gains [16]. Another emblematic example is the use of RL to find out how birds find and navigate ascending thermal plumes in the atmosphere as they search for prey or migrate [80]. Here by learning the optimal policy, the authors were able to find what are the mechanosensory cues that help birds guide themselves in this scenario. Other successful implementations of RL include, real-time traffic signal control [1], optimizing energy conservation and comfort in buildings [20] and autonomous helicopter flight [71] among others.

Dynamic surrogate-modeling optimization The RL framework descri-bed above although represents a promising method could be computational prohibited [38]. In this context, the use of a dynamic surrogate model that can provide fast approximations of the reward or objective function is an interesting alternative. Furthermore, if this surrogate can also provide a reasonable prediction of time-varying key performance variables, e.g., water rates, oil rates, it could potentially replace the numerical reservoir simulator altogether. Once a validated surrogate model has been identified it can be coupled with any optimization algorithm (*e.g.* genetic algorithms, RL) with relative ease and low computational effort.

A common approach for dynamic surrogate-modeling [7], [43], [54], [95], is to use nonlinear models to make one-step forecasts considering the current state of the process and current action control as inputs and the state at the following time step as output. Afterward, the model is used recursively to make n-steps ahead forecast: the predicted state at a time t, is used as input, to predict the state at $t+1$. However, the main limitation of this approach is that prediction errors obtained after the initial time steps are propagated throughout the entire prediction horizon. In other words, the error is expected to grow as the number of forecast steps increases.

1.3 Research Objectives

This research aims at evaluating and proposing state-of-the-art machine learning-based approaches to solve the optimal control problem expressed by Equation 2.1. This goal can be expressed in the following specific research objectives:

1. Find optimal steam injection policies for different SAGD reservoir simulation models (case studies) using reinforcement learning. Case studies differ in the number of well pairs, considered and RL approach used, *e.g.* value function and policy gradient. For each case, this work will include, definition of state and action space, function approximation strategy and parameter tuning. Furthermore, optimal policies will be analyzed in two aspects:

    1.1 Optimization or learning process. In particular, we are interested in the agent's improvement after each iteration or episode (learning curve), optimal design variables

    1.2 Qualitative analysis of the optimal steam injection policies to gain insight into the SAGD process, *e.g.* what role does pressure play throughout the production horizon? Is high pressure more important at the beginning?

What role does temperature play? Is it important to keep a high temperature throughout the production horizon or towards the end?

2. Evaluate a recursive prediction approach using one-step machine learning-based forecast models for SAGD optimization. This implies the estimation of key time-varying output, *i.e.* oil production rate, water production rate, net present value, using a numerical simulation model of a multiwell SAGD process as case study. This objective comprises the following four (4) stages: design of experiments, numerical simulations of each sample, surrogate model identification and validation and optimization. Furthermore, surrogate model identification will require the definition of, an effective feature set, model architecture, training and validation methodology, and perform a statistical evaluation of the results

3. Develop a dynamic surrogate model methodology to overcome the error propagation limitation of the conventional recursive approach evaluated in **Specific Objective 2**. This includes the four (4) stages stated above, plus an additional stage that requires the definition of a secondary model to capture the dynamics of the residuals exhibited by the recursive model. The methodology is tested on a numerical simulation model of a multiwell SAGD process and a statistical evaluation of the results is given

1.4 Key Contributions

The completion of the cited objectives will make the following contributions:

- Formulate the problem of finding the optimal steam injection policy for SAGD process as an optimal control problem (**General**)

- Implement and evaluate action value and policy gradient algorithms (SARSA and REINFORCE, respectively) to solve the cited optimal control problem (**Specific Objective 1**)

- Interpret obtained optimal steam injection policies on the basis of its physical relevance in the context of the SAGD process, *e.g.* role of temperature and pressure (**Specific Objective 1**)

- Develop an efficient and effective methodology for the construction of dynamic surrogate models for rapid evaluations of SAGD steam injection policies (**Specific Objective 2 and 3**)

- Apply the developed methodology to case studies with varying number of well pairs (**Specific Objective 2 and 3**)

1.5 Thesis Outline

The organization of this work is organized in seven (7) chapters that are briefly described below:

Chapter 2: Literature review Describes the three (3) main methods for solving the optimal control problem cited in Equation 2.1, namely, adjoint-based optimization, MPC and RL. In particular, discusses the two main RL approaches (action-value function and policy gradient) and reviews the work done in the application of reinforcement learning strategies for the optimization of reservoir operations, *e.g.* waterflooding, CO2 storage. Additionally, contains a detailed review of the dynamic surrogate modeling techniques and offers the main previous studies of their applications for the optimization of subsurface operations

Chapter 3: Action-value function RL for SAGD optimization Pre-sents the implementation of the on-line on-policy action-value function algorithm SARSA for the optimization of a one-well pair SAGD reservoir simulation model. The implementation considers a discrete action space, a continuous state space and Net Present Value (NPV) is used as a reward/objective function. Additionally, the action-value function is approximated using a stochastic gradient regression strategy, and the state vector is featurized using radial basis kernels. Results are evaluated on the basis of improvement over iterations (or episodes) and on a justification of the optimal policy from a physical point of view

Chapter 4: Policy gradient RL for SAGD optimization Contains the details of the implementation well-known policy gradient algorithm REINFORCE for the optimization of a multi-well pair SAGD reservoir simulation model. The implementation considers a discrete action space, a continuous state space and Net Present Value (NPV) is used as a reward/objective function. Additionally, the policy is parametrized by a deep neural network that offers for every state the probability of taking a specific action. Results are compared to a constant injection approach and evaluated on how the objective function is improved as the number of iterations is increased, and a physical interpretation of the optimal policy is offered

Chapter 5: An evaluation of the recursive prediction approach for dynamic surrogate modeling and optimization Offers an evaluation of using one-step forecast machine learning-based models in a recursive fashion to make n-step prediction of time-varying outputs of a SAGD process. These models are identified offline and then used as a substitute for the numerical reservoir simulation model, considered computationally expensive, in the optimization process. The approach is applied to a multi-well reservoir simulation model, and the performance of the approach is evaluated in terms of its effectivity

Chapter 6: A corrected recursive based dynamic surrogate model approach Proposes an effective surrogate-based approach in which, the prediction at any given time consists of the addition of two components: a one-step forecast nonlinear models used recursively to make n-steps ahead forecast plus a modeled residual correction. The latter is rationalized under the assumption that the forecast error given by the one-step forecast model is correlated with time due to the recursive strategy used, as a consequence the residual can be modeled. Results are compared to the conventional recursive-based approach on the basis of, the reduction of mean errors and increase in R2.

Chapter 7: Conclusions and recommendations The main conclusions and findings of the research are summarized in this chapter and potential future research directions are discussed.

Chapter 2

Literature Review

In this chapter, we briefly describe the key SAGD optimization performance measures typically used in the field and the formal optimization problem is presented. Also we present the static and dynamic approaches (Figure 2.1) for the optimization problem of interest. In particular, the main methods of each approach are discussed, i.e., gradient-based and heuristic methods in terms of static optimization, and adjoint-method, model-predictive control (MPC) and reinforcement learning (RL) regarding dynamic optimization. Moreover, the main applications of these methods for hydrocarbon recovery strategies are offered. Additionally, we present the conventional surrogate base model optimization (SMO) methodology as a possible solution to the cited approaches and their previous applications in oil recovery methods.

2.1 SAGD Performance Measures

There are two (2) performance measures that are typically considered in SAGD optimization: achieving a uniform expansion of the steam chamber along the length of the well pair (conformance) and maximizing net present value (NPV). The first case is associated with real time operational performance and typically achieved by controlling the subcool temperature: temperature difference between the saturation temperature and the bottom hole temperature of the produced water. In the second case (NPV), performance is seen from a long-term perspective and the economical aspect of the process is considered, *i.e.* operating costs, oil prices, water produced. Regardless of the performance measures the key design variables of interest are: injecting pressure or rate, and producing pressure at every time step.

Steam chamber conformance requires subcool control stragies to ensure a liquid pool above the lower producer well, also known as a steam trap [24], [114]. Liquid pool is defined as the condensed or mobilized fluid near the producer well that flow down the edged of the steam chamber. On one hand, if the liquid pool is too low (low subcool temperature, high production), steam is quickly condensed and produced, thus decreasing the thermal efficiency of the process: heat is not heating the surrounding oil [35]. On the other hand, if the liquid pool is too high (high subcool temperature, low production) decreases oil production.

Although both subcool control and NPV optimization consists of selecting the values of the operating conditions (*e.g.* steam injection rates, prouction pressure), they work on different time scales. Subcool control is considered o a scale of days to month and is typically achieved using control strategies such as, Proportional-Derivative-Integral (PID) control, and Model Predictive Control (MPC). While NPV optimization is considered a field development planning strategy, in a scale of years.

In this work, we are interested in NPV optimization and the problem formulation is as follows:

> For a given SAGD configuration with n injector and producer wells (pairs) and a given production horizon with T time steps (*e.g.* daily), the objective is to find the steam injection policy that maximizes a particular performance measure (*e.g.* net present value) while accounting for the complex reservoir dynamics. Formally, the optimization problem can be written as:

$$\begin{aligned} \max_{\mathbf{u} \in \mathbb{R}^{n \times T}} \quad & J = \sum_{t=1}^{T} J_t(x_t, u_t) \\ \text{s.t.} \quad & \mathbf{u}_{min} \geq \mathbf{u} \geq \mathbf{u}_{max} \\ & x_{t+1} = \mathbf{g}(x_t, u_t) \end{aligned} \tag{2.1}$$

where J is a scalar performance measure, $u_t \in \mathbb{R}^{n \times 1}$ is the steam injection rates for all n wells at time t, x_t is the state matrix of the SAGD dynamic system (*e.g.* pressure and saturation distribution) at time t, $\mathbf{u}_{min}$ and $\mathbf{u}_{max}$[1] are the upper and lower bounds of the steam injection rate, and $\mathbf{g}$ is a function that represents the SAGD dynamic system

2.2 Static Optimization

This approach refers to the case where the design variable (*i.e.* steam injection rate) at each time step (u_t) is assumed as an independent variable. In this case, the constraint function g that represents the SAGD dynamical system is assumed to be a black box, *i.e.* the system is available for objective function evaluations but the equations that describe the system are unknown. There are two main groups of methods used for this approach, gradient-based and heuristics methods.

2.2.1 Gradient-based

These are represented by a family of algorithms in which some form of the gradient of the objective function J is used to estimate a descent direction. The general structure of these problems is that one starts at an initial point ($\mathbf{u}_k$), estimates a descent (ascent) direction according to some fixed rule, and then takes a step (α) in that direction [57]. Mathematically, this structure can be expressed as,

[1]To clarify, the bold and non-italic representation of $\mathbf{u}$ express that contains all the steam injection rates for each well at each time step.

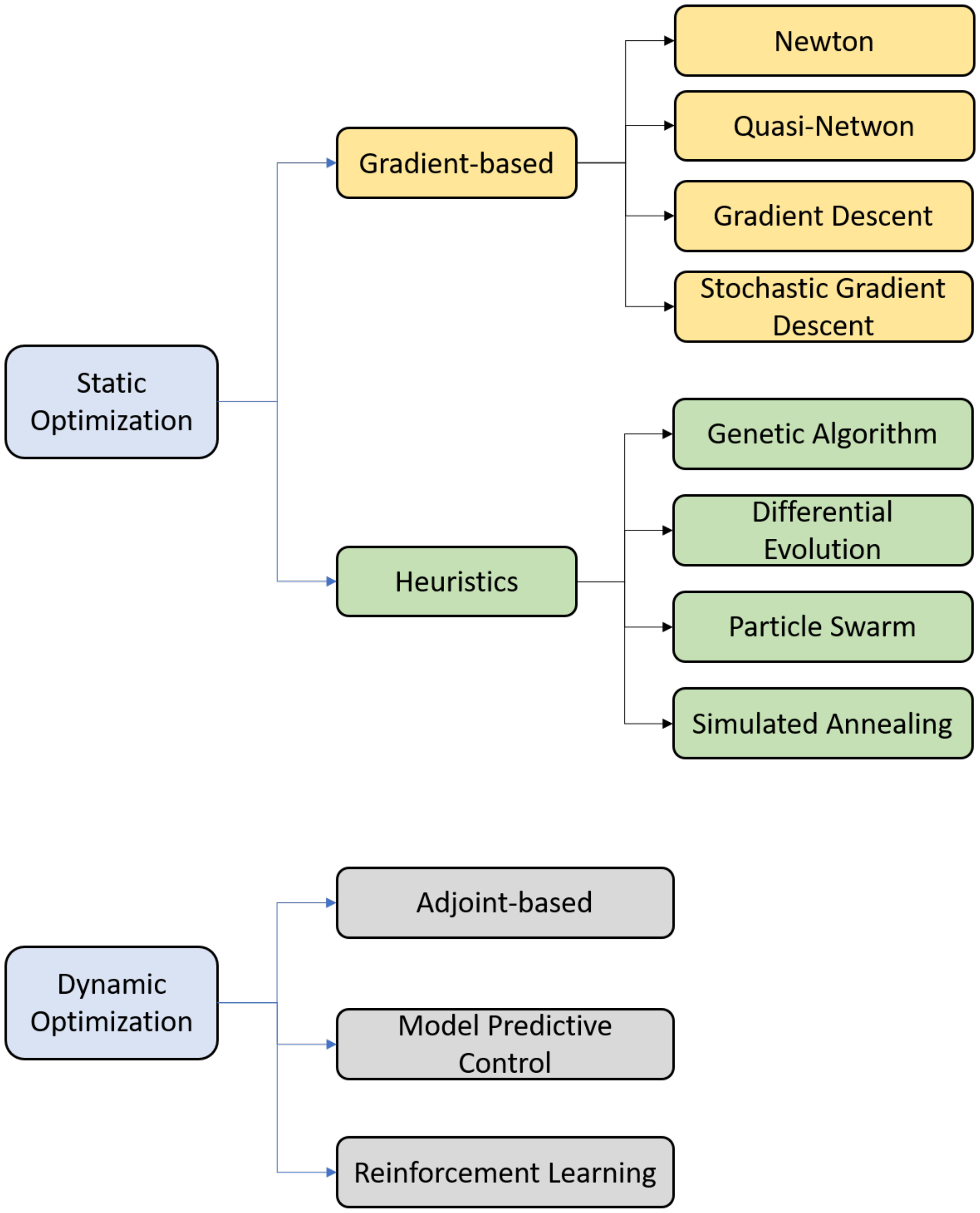


Figure 2.1: Methods for static and dynamic optimization

$$\mathbf{u}_{k+1} = \mathbf{u}_k + \alpha f(J(\mathbf{u}_k)) \tag{2.2}$$

Descent direction

The main difference between the different gradient-based methods is how to estimate the descent direction $f(J(\mathbf{u}_k))$. The simplest method is the Steepest Descent that uses the negative of the gradient of $J(\mathbf{u}_k)$, *i.e.*

$$\mathbf{u}_{k+1} = \mathbf{u}_k - \alpha \nabla J(\mathbf{u}_k) \tag{2.3}$$

More sophisticated methods include Newton and Quasi-Newton, in which information of the second derivative or Hessian is used to estimate a new descent direction. For example, the Newton method uses the full Hessian, *i.e.*

$$\mathbf{u}_{k+1} = \mathbf{u}_k - \alpha [\nabla^2 J(\mathbf{u}_k)]^{-1} \nabla J(\mathbf{u}_k) \tag{2.4}$$

The Quasi-Newton methods use approximations of the Hessian, the most common method is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm which uses the procedure described in Algorithm 1.

```
Algorithm 1: BFGS
  From initial guess $\mathbf{u}_0$ and $B_0 = I$
  1. Estimate the descent direction $\mathbf{p}_k$ as $B_k\mathbf{p}_k = -\nabla J(\mathbf{u}_k)$
  2. Obtain step size $\alpha_k$ by $\arg\min_\alpha(\mathbf{u}_k + \alpha_k\mathbf{p}_k)$ - See following section
  3. Set $\mathbf{s}_k = \alpha_k\mathbf{p}_k$
  4. Estimate a new solution as, $\mathbf{u}_{k+1} = \mathbf{u}_k + \mathbf{s}_k$
  5. $\mathbf{y}_k = \nabla J(\mathbf{u}_{k+1}) - \nabla J(\mathbf{u}_k)$
  6. Update $B_{k+1} = B_k + [\mathbf{y}_k\mathbf{y}_k^T] \cdot [\mathbf{y}_k^T\mathbf{s}_k]^{-1} - [B_k\mathbf{s}_k\mathbf{s}_k^T B_k^T][\mathbf{s}_k^T B_k\mathbf{s}_k]^{-1}$
```

In all cases, the underlying assumption of any gradient-based optimization algorithm is that the objective function is differentiable and convex. Typical implementations rely on the numerical approximations of the gradient, hessian or approximations thereof. These implementations include sklearn.optimize (Python) and fminunc/fmincon (Matlab).

Step size - α

The process for determining the step size (α) or how much to move in the descent direction is called line search. This represents a one-dimensional minimization (maximization) problem and can be expressed as,

$$h(\alpha) = \mathbf{u}_k + \alpha f(J(\mathbf{u}_k)) \tag{2.5}$$

Here the goal is to find the optimal α that will minimize h. A common method for line search is the cubic fit; here, three (3) points are initially selected using a fixed heuristic and then used to fit a cubic function with a known minimum (Figure 2.2). Afterward, the minimum replaces one of the previous three points and a new cubic function is fit. This process is repeated until a predetermined stopping criterion is met.

Other line search methods include the golden section, grid search, secant method, etc. Additionally, in other settings, such as reinforcement learning and supervised learning α is referred to as the learning rate.

2.2.2 Heuristics

These methods imply the use of rules or guidelines, typically nature inspired, to find good solutions. Heuristics or non-gradient-based algorithms are an alternative in cases where the objective function is not differentiable (*e.g.* discrete design variables) or approximations of the gradient of the objective function are not feasible. As a consequence, these strategies offer no sort of guarantee to find global or bounded

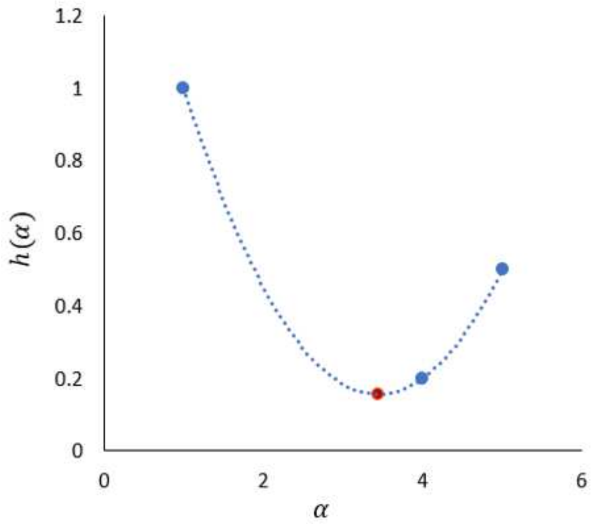


Figure 2.2: Illustration of the line search process. The blue points represent the initial set, the dashed blue line the cubic fit, and the red point the optimal of the cubic function

optimal solutions [102]. Moreover, similarly to gradient-based methods discussed above, the assumption is that the objective function is considered to be a black box.

In this section four (4) heuristics optimization methods will be discussed: two (2) evolutionary-inspired methods: genetic algorithms and differential evolution, as well as the particle swarm and simulated annealing methods.

Genetic algorithms

Genetic algorithms are non-gradient optimization algorithms in which initial solutions are evolved to optimal solutions by biologically inspired operators, *e.g.* crossover, mutation. Each solution is coded in binary format and referred to as *genes*.

In general, the algorithm consists of three (3) stages, initialization, creating a next generation and termination. Initialization refers to starting with an initial population or a predetermined number of candidate solutions, typically generated randomly. Each of these candidate solutions is evaluated and the solutions are ranked according to their corresponding performance. The population is referred to as a *generation* and as the iterative process continues new generations are generated with higher performance.

In the second stage, the new generation is created in three (3) ways,

- Elitism. Best individuals are selected to continue on to the following generation
- Crossover. These individuals are a result of the combination of two (2) good performing parents. The offspring will share characteristics of the parents and will tend to have a better performance or NPV value.
- Mutation. Implements random changes to selected individuals

These two stages are repeated until a termination condition is achieved and the algorithm is terminated (third stage). These conditions may include, evolving for a pre-determined number of generations or iterations and the NPV values of the best-performing individuals do not differ significantly from generation to generation.

Differential Evolution

Consists of creating generations or groups of samples and then iteratively selecting new solutions that will exhibit a higher objective function value [96]. Unlike genetic algorithms, each solution is represented by floating numbers. Initially, the first generation is chosen randomly and then three (3) operators are sequentially applied:

1. Mutation. The weighted difference between two samples is added to a third sample to produce a mutated sample
2. Crossover. The parameters of the mutated sample are then randomly mixed with the parameters of another sample
3. Selection. The best samples are selected for the next generation

This process continues until convergence has been reached, *e.g.* maximum number of generations.

Particle Swarm

Inspired by swarm intelligence, mimics the behavior of groups of species such as bird flocking, in attempts to find shelter or food. In the search, each individual will attempt to balance their own knowledge (exploitation) and of the flock (exploration).

The method consists of A_p individuals or particles exploring the search space. Each individual knows their own position and velocity and will adjust to a new position based on, the best position visited by itself and the best position visited by any other member of the flock. This update consists of the addition of three (3) components:

- The previous velocity multiplied by an inertia weight that represents a trade-off between exploration and exploitation
- A cognitive learning factor that denotes how much of the individual's own success will impact the new velocity
- A social learning factor that embodies the attraction toward the flock's success

Simulated Annealing

Inspired by the annealing technique in metallurgy that consists of heating and controlled cooling of a material to form a strong and required crystalline structure. This will only be achieved by controlling the initial temperature and cooling schedule.

The basic algorithm starts with an initial state, which represents the initial guess of the algorithm, and an initial temperature T_{sa}. At each iteration, a random neighbor ($\mathbf{u}'$) is generated, if the new neighbor improves the objective function, the current state transitions into the new state. If not, the neighbor is selected with a given probability distribution that depends on the current temperature and the objective function deterioration (ΔE). This probability is expressed by the Boltzman distribution:

$$Pr(\Delta E, T_{sa}) = \exp^{-\frac{J(\mathbf{u}') - J(\mathbf{u})}{T_{sa}}} \tag{2.6}$$

When a new state is accepted or an equilibrium state is reached, the temperature T_{sa} is gradually decreased according to a predetermined schedule, *e.g.* linear, geometric, logarithmic.

Heuristics optimization methods for the optimization of enhanced oil recovery methods are implemented in the CMG reservoir simulation package as CMOST. In particular, differential evolution and particle swarm are available.

2.3 Dynamic Optimization

Also referred to as optimal control or trajectory/path optimization, dynamic optimization deals with finding an optimal policy or control actions of a dynamical system. Unlike the static optimization approach, the time component of the control actions is accounted for.

2.3.1 Adjoint-based optimization

Even though also represents a gradient-based approach, adjoint- based optimization is used when the system's dynamic is described by a set of known partial differential equations. In particular, it represents an efficient way to compute gradients [48] of the objective function with respect to the control variables. These gradients are found by applying the Karush-Kuhn-Tucker conditions to the Hamiltonian, *i.e.*, an augmented objective function represented by the pre-established performance measure (*e.g.* net present value) and the constrained imposed by the reservoir dynamics (*e.g.* mass-conservation partial differential equations) using Lagrange multipliers.

Assuming no other constraints except for those imposed by the physical system ($\mathbf{g}$), the optimization problem given by Equation 2.1 can be formulated as:

$$\begin{aligned} \max_{\mathbf{u}} \quad & J \\ \text{s.t.} \quad & \mathbf{g}(u_t, x_t) - x_{t+1} = 0 \end{aligned} \tag{2.7}$$

where x_t and u_t represent the vector of state variables (*e.g.* pressure distribution) and the control actions at time t. The constrained optimization problem can be transformed into an unconstrained one using the Hamiltonian function as a new augmented objective function, $\bar{J}$, *i.e.*

$$H_t(x_{t+1}, x_t, u_t, \lambda_{t+1}) = J_t(x_t, u_t) + \lambda_{t+1}^T \mathbf{g}_t(x_t, u_t) \tag{2.8}$$

where λ represent the Lagrange multiplier(s). Applying the Karush-Kuhn-Tucker conditions to Equation 2.8 gives rise to the state equation, adjoint system, and stationary conditions that when solved the gradient of the objective function with respect to $\mathbf{u}$ is found.

$$x_{t+1} = \frac{\partial H_t}{\partial \lambda_{t+1}} = \mathbf{g}_t(x_t, u_t), \quad \text{State Equation} \tag{2.9}$$

$$\lambda_t = \frac{\partial H_t}{\partial x_t} = \left(\frac{\partial \mathbf{g}_t}{\partial x_t}\right)^T \lambda_{t+1} + \frac{\partial J_t}{\partial u_t}, \quad \text{Co-state Equation or Adjoint System} \tag{2.10}$$

$$0 = \frac{\partial H_t}{\partial u_t} = \left(\frac{\partial \mathbf{g}_t}{\partial u_t}\right)^T \lambda_{t+1} + \frac{\partial H_t}{\partial u_t}, \quad \text{Stationary Condition} \tag{2.11}$$

The state equation (Equation 2.9) is the dynamic system's equation which is solved in a forward fashion, *i.e.* from x_t, u_t to x_{t+1}. The co-state equation (Equation

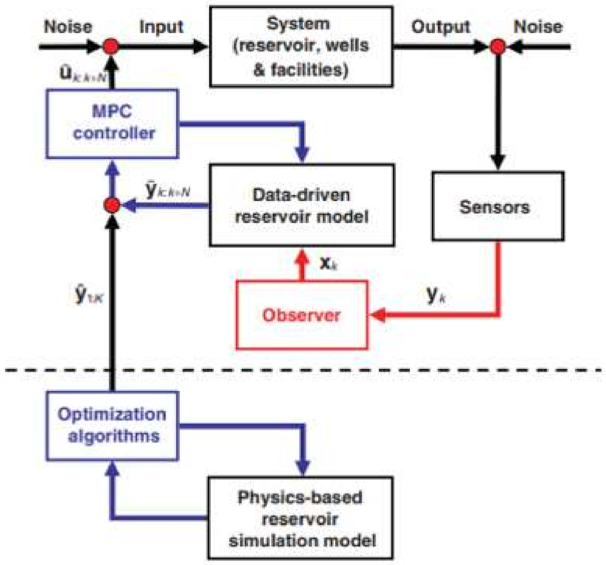
2.10) or adjoint system represents a different dynamic system which is solved in a backward fashion, *i.e.* λ_{t+1} is used to find λ_t. Finally, Equation 2.14 is used to obtain the optimal values of $\mathbf{u}$ by knowing through $\frac{\partial H}{\partial \mathbf{u}}$ and any gradient-based optimization method (See Section 2.2.1).

This approach has been implemented in the well-known reservoir simulation software Eclipse 300 (Schlumberger) [86] as the keyword OPTIMIZE. As a by-product of the adjoint-based method, the software also offers sensitivity calculations of the objective function with respect to the control variables using the obtained gradients.

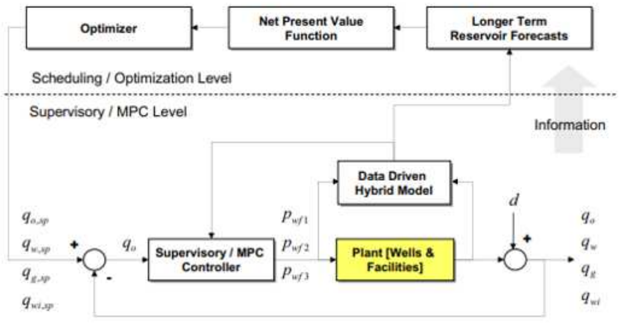
The main advantage of this approach is that its computational expense is independent of the number of design variables (*e.g.* number of considered well pairs). Thus, we can find successful implementations in a variety of oil recovery processes to find the optimal policy of operating conditions (*e.g.* injection rates, production pressure). These implementations include, CO2 water-alternating-gas injections [13], polymer flooding [104], co-optimization of CO2 and oil recovery [47], waterflooding [105] and well-placement scenarios [116]. However, the biggest drawback is that the gradients of the dynamic system with respect to x and $\mathbf{u}$, must be hard-coded and are different depending on the enhanced oil recovery process.

2.3.2 Model-based predictive control (MPC)

Typically consists of a two-level control strategy [25], [39], [85], *i.e.* an *upper level* or optimization level, which is responsible for finding the set-points: injection (*e.g.* water, steam) and oil production rates that will maximize net present value of a determined production horizon; and a *lower level* or MPC level, which includes the MPC controller, that must find the operation conditions (*e.g.* bottom-hole pressures) to reach the set-points given by the upper level. Figure 2.3 shows the block diagrams corresponding to two different MPC implementations for the solution of the optimal control problem.



(a)



(b)

Figure 2.3: Block diagram of the MPC-based framework proposed by [85]-top and [25]-bottom

Note in Figure 2.3 that in the upper level, the choice of how to generate the set-points can be quite different. So for example, in [25]-Figure 2.3b a full physics-based reservoir simulation model along with an adjoint-based optimization framework (see 2.3.1) is used. Despite having the cited drawbacks, this guarantees a maximization of cumulative net present value for the entire production horizon T. In contrast, in [85]-Figure 2.3a linearized versions of the first principle analytical equations (Vogel and Fetkovitch for well flow, and Havlena-Odeh for pressure) were used. Another possibility is to use data-driven models identified online as new data is available. The choice of model (*e.g.* first principle or data-driven) leads to a subtle difference from the objective function in Equation 2.1, i.e.,

$$\max_{\mathbf{u}} \quad J = \sum_{z=1}^{Z} J_t(x_h, \mathbf{u}_h) \tag{2.12}$$

where Z is not necessarily equal to T, depending on the capacity of the model used to forecast in the future. Regardless of the model being used, the objective function for the upper level is typically the same, *i.e.*

$$\sum_{t=1}^{T} \left\{ \frac{\sum_{j=1}^{n} [P_o \cdot q_{o,j,t} - C_{water} \cdot q_{w,j,t}] - \sum_{j=1}^{n_i} [C_{steam} \cdot q_{s,j,t}]}{1 + i^{\frac{t}{365}}} \right\} \tag{2.13}$$

where, T is the full production horizon (*e.g.* 20 years), n_p is the number of production wells, P_o is the oil price [USD/STB], $q_{o,j}$ the oil production rate of the j-th well [STB/day], C_{steam} the cost of steam generation [USD/STB], $q_{s,j}$ the steam injection rate of the j-th well [STB/day], C_{water} the cost of produced water handling [USD/STB], $q_{w,j}$ the water production rate of the j-th well, n_i the number of injection wells, i is the annual discount factor [fraction] and t is the current time.

The lower level consists primarily of the MPC controller and the system or plant (*e.g.* reservoir, wells, facilities). In general, the controller's job is to find the steam injection rates to achieve the set-points - $\tilde{ym}$ (*e.g.* oil production rates) offered by the upper level by interaction with the plant or system and solving the following optimization problem,

$$\min_{\tilde{u}_{k:k+N}} \quad \sum_{j=k}^{k+N} (\tilde{ym}_j - \hat{ym}_j)^T \mathbf{W}_1 (\tilde{ym}_j - \hat{ym}_j) + (\Delta u)^T \mathbf{W}_2 (\Delta u) \tag{2.14}$$

where, $\hat{ym}$ is the outputs offered by the plant, Δu is the change in steam injection rates and N is the moving optimization horizon, k is the current time step, and $\mathbf{W}_1$ and $\mathbf{W}_2$ are weighting matrices. The first part of the right hand side represents the difference between the desired and the predicted output, while the second part aims to penalize big changes in u. Although the mechanics of this lower level is very similar, they can typically differ in terms of the choice of the data-driven model (*e.g.* neural networks, system identification approaches) and the design variables.

Note how the time scale in the lower and upper level is not necessarily the same ($T \neq Z \neq N$). In fact, the upper level is expected to have a longer horizon in order to account for a more long-term optimization, this is known as the optimization horizon. In contrast to the adjoint-based optimization, where the optimal control problem is solved once over the entire production horizon (*i.e.* 20 years), MPC solves the problem repeatedly as the optimization horizon is moving.

MPC has proven to be a very effective approach in that, it allows for real-time optimization and doesn't necessarily implies the use of a complex model for its upper level. In this regard, we can find literature on different variants of MPC, such as, linear [75] and nonlinear MPC [63], [75]. Additionally, effective implementations in the field of oil recovery optimization include, enhancing fracture surface area in naturally fractured reservoirs [93] and SAGD [107]. However, the drawback is that the nature of model (used to find the set-point values), may be unfeasible for long-term cumulative optimization of net present value.

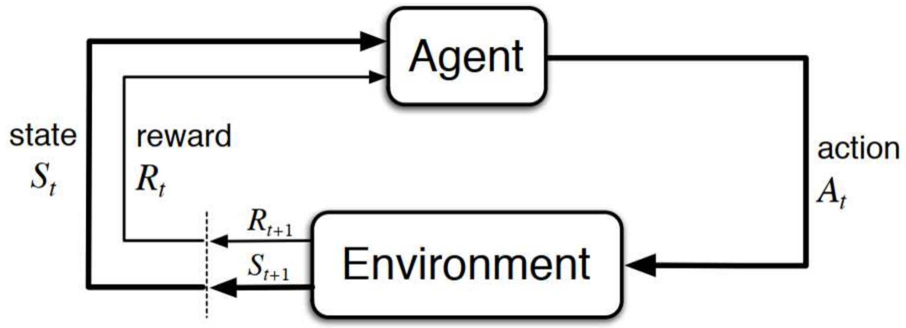


Figure 2.4: Agent-environment interactions in reinforcement learning [99]

2.3.3 Reinforcement learning (RL)

In RL, an agent is trained to find a *policy* that will maximize total future reward only by continuous interactions with the environment. A policy π is defined as the behaviour or action a of the agent at each state s of the environment. The policy is pre-defined and is continuously improved during the training process. Mathematically a deterministic policy can be expressed as:

$$a = \pi(s) \tag{2.15}$$

The agent is expected to find the optimal policy with no prior knowledge of the dynamics of the environment. At each time step, the agent executes an *action* (e.g., increase steam injection rate), receives a *reward* (e.g., net present value), and receives a representation of the new *state* (e.g., pressure distribution) of the *environment* (e.g., numerical reservoir simulation model) as can be seen in Figure 2.4. Note that this represents a major difference from traditional machine learning approaches such as supervised learning in which the training data set describes the correct action the system should take for a given input; then, the agent must extrapolate or generalize its response so that it acts correctly in situations not present in the training set.

In this context, a *reward* R_t represents a scalar feedback signal that indicates its performance at time step t. Some examples of reward include, for a fly stunt manoeuvres in a helicopter a positive reward for following a desired trajectory and negative for crashing, in a power station a positive reward for producing power and negative for exceeding safety threshold. In RL we are interested in finding the actions so that the sum of the discounted rewards it receives over a specific horizon T (episodic applications) is maximized. This is also called the expected discounted *return* and can be expressed as:

$$J = G_t = R_{t+1} + \gamma R_{t+2} + ... = \sum_{k=1}^{T} \gamma^k R_{t+k+1} \tag{2.16}$$

where, γ is a number between 0 and 1 and determines the present value of future rewards, so for example, if $\gamma = 0$ the agent is only concerned with maximizing immediate rewards (myopic). If γ approaches 1, more importance is given to future rewards (far-sighted). In the context of the optimization problem, G_t is equivalent to J (Equation 2.1).

The *environment* is formulated as a Markov decision process (MDP) in which the defined *state* at a particular times_t captures all relevant information from the history. In other words, the current state captures all the information from the past and is enough to transition to the next state as a result of a specific action. For example, if we were interested in maximizing the energy output of a thermoelectric power station, the state could be the pressure and temperature reading of the boiler at a given time. In each *state* the *action* space $\mathcal{A}$ is represented by the set of actions the agent can take. In the power station, actions could include, increase fuel and air mass flow rate at the inlet of the combustion process. The *transition* from state to state is given by the dynamic of the world (e.g., combustion dynamics in the boiler of the power plant) which is assumed unknown and potentially stochastic. Furthermore, the *environment* is considered a black box and no attempt is made to find the model describing its dynamics.

There are two main strategies in reinforcement learning, i) action-value and ii) policy gradient methods. In the former, an action-value function is learned - $\hat{q}_\pi(s, a, \mathbf{w})$; this function offers the expected return starting from state s, taking action a, and thereafter following policy π, implementations of this strategy include Q-learning and SARSA. In this case, the policy is not explicitly given; the actions are selected based on the estimation provided by the action-value function. So, for example a greedy policy would be,

$$\pi(s|a) = arg\max_a \hat{q}_\pi(s, a, \mathbf{w}) \tag{2.17}$$

where $\mathbf{w}$ represents the weights that parametrize the approximate action value function $\hat{q}_\pi$. In the latter, a parametrized policy is learned - $\pi(s|a, \theta)$; the actions are selected by simply evaluating π without necessarily using value function. Here, θ are the parameters of learned policy and can correspond for example to the parameters of a deep neural network. Policy gradient algorithms include, Monte Carlo and Proximal Policy Optimization (PPO).

Both of these strategies have been previously used for the selection of injection rates in various oil reservoir operations applications. In particular, for waterflooding, Q-learning using a function approximation approach with deep neural networks - DQN [58], a Proximal Policy Optimization implementation [64], and a basic form of an averaging method as an update rule [45]. Additionally, for carbon storage, DQN was used to maximize the amount of CO_2 while minimizing potential risks [97]. Also for waterflooding, a policy gradient approach was applied for the life-cycle production optimization [117]. Other RL applications in the context of the oil and gas industry are guided drilling control using SARSA and deep neural networks [55], finding an optimal well placement and well type for subsequent wells using deep-q-networks - DQN [21] and dynamic scheduling of maintenance tasks in refinery production systems using SARSA [2]. Implementations of both the action-value and

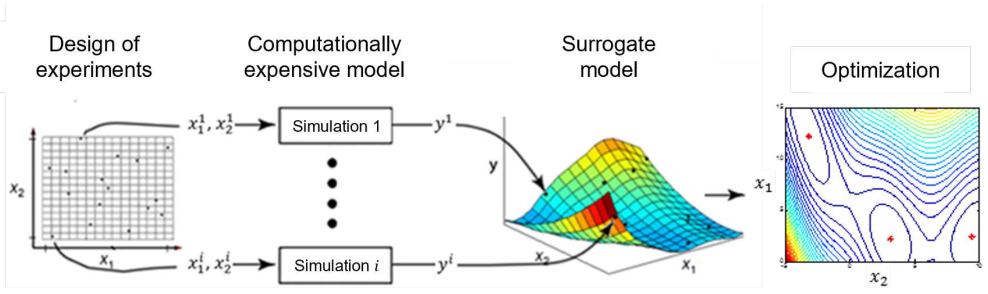


Figure 2.5: Illustration of the surrogate modeling and optimization framework

policy gradient approach for SAGD optimization can be found in **Chapter 3** and **4**, respectively.

2.4 Surrogate Modeling

Except for MPC and adjoint-based optimization, in the rest of the discussed methods multiple evaluations of the objective function are necessary to find optimal solutions. A straightforward option is to directly use a high-fidelity reservoir simulation model, however due to the complexity of the SAGD process (as explained Chapter 1) the computational cost of each evaluation could prove to be prohibitive.

To alleviate this computational burden generating fast models that can be used as reasonable proxy of the high-fidelity model [61], [79] is a common practice. To this end, there are two basic approaches for this: i) simplified or approximated physics [28], [106] and ii) data-driven or black-box models [49], [77], [78].

2.4.1 Physics-based approach

Consists of using analytical or simplified physics models that, although are based on ideal assumptions that simplify the physical process but capture the main mechanisms that drive the process. For SAGD, this could be done [106] using solutions based on the Butler SAGD theory [9], [81] to make predictions of cumulative oil production, steam chamber, etc. Furthermore, the Butler SAGD model can be modified to represent a solvent injection scheme for heavy oil recovery (VAPEX) [88], [89] to predict total drainage rate and he advancement of solvent chamber.

2.4.2 Data-based approach

Also called black-box modeling, consists of using entirely input-output data obtained from a high-fidelity model to identify a surrogate or proxy model.

Surrogate modeling and optimization (SMO)

In the typical SMO approach, the goal is identify a continuous function from a limited amount of available data (samples or experiments). This data is usually generated using a computationally expensive model that represents the objective function and/or the constraints of the optimization problem. Here, input variables are typically non-ordered, do not change over time and are, to some extent, unrelated to one another. In particular, this approach [78] consists of four (4) main stages (Figure 2.5):

1. *Design of experiments.* Generate m representative samples from the design variable space

2. *Evaluation of each sample using a high fidelity model.* Consists of obtaining a corresponding output value for each of the i-th samples from a computational expensive model

3. *Surrogate model identification and validation.* Find a proper structure and the corresponding parameters that can map the input (Stage 1) and output (Stage 2). Represents an inverse problem, given that multiple structures/parameters may be consistent with the data (Figure 2.6) Validation refers to checking the

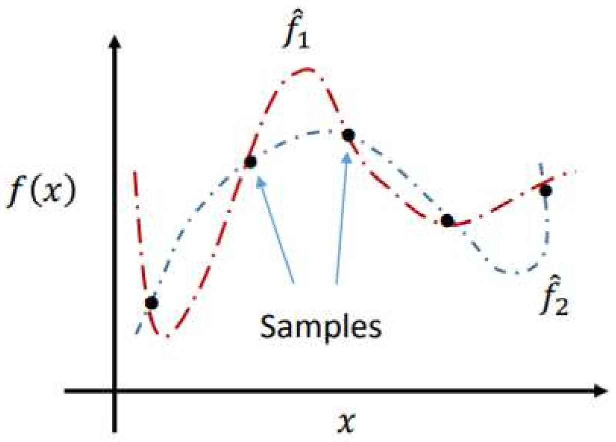


Figure 2.6: Illustration of the inverse problem of surrogate model identification

generalization capacity of the surrogate, usually using data that was not used in the identification process.

4. *Optimization.* Find the value of the input variables that can maximize the output or a function that depends thereof

In the oil recovery setting, common applications of SMO include finding optimal values of, well placement [67], [84], [98], water alternating gas (WAG) cycle ratio [4], the concentration of alkaline, surfactant, and polymer (ASP) in injection [11], vertical well spacing and steam injected enthalpy in SAGD operations [77], waterflooding optimization [14], [15], [44], [73], [113], [115].

Time series modeling

This is a setting [8], [54], [103] in which historical time series data is available and the goal is to predict n-steps in the future as illustrated in Figure 2.7. Moreover, to predict future steps, *autocorrelation* [59] is expected, *i.e.* future values of x dependent on past endogenous data (previous values of x) and can include exogenous data, *e.g.* external actions. In general, the goal is to model the prediction of future steps as:

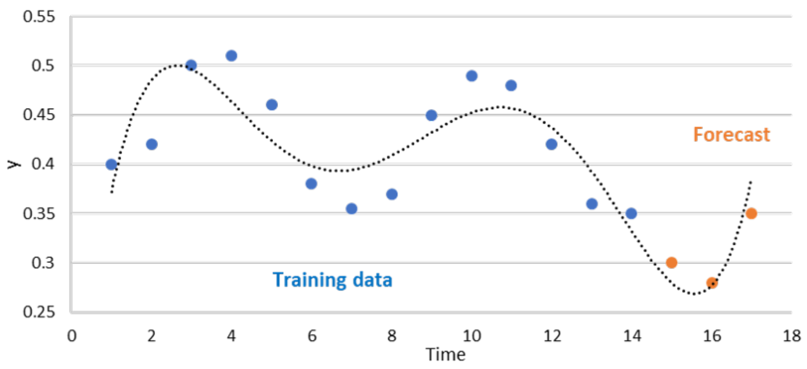


Figure 2.7: Illustration of time series modeling and forecast from training or historical data

$$z_t = \mu + e_t + \Psi_1 a_{t-1} + \Psi a_{t-2} + ... \qquad (2.18)$$

where, μ is the mean of the process, z_t prediction interest, e are a sequence of random variables that represent the white noise process and Ψ are the parameters of the model. Equation 2.18 is called the linear filter model and represents the general form of linear time series modeling. Special cases that also include exogenous input,

- Autoregressive (ARX) models

$$\bar{z}_t = e_t + \sum_{i=1}^{p} \tau_i \bar{z}_{t-i} + \sum_{i=1}^{b} \phi_i u_{t-i} \qquad (2.19)$$

  where $\bar{z}_t = z_t - \mu$, τ and ϕ are the parameters of the model corresponding to the autoregressive and exogenous output component, p and b represents the order of the model for both components and u is the exogenous input (*e.g.* steam injection rate)

- Moving average (MAX) models

$$\bar{z}_t = e_t + \sum_{i=1}^{q} {}_d\psi_i e_{t-i} + \sum_{i=1}^{b} \phi_i u_{t-i} \tag{2.20}$$

  where q_d represents the order of the moving average component of the model

- Mixed autoregressive-moving average (ARMAX) models

$$\bar{z}_t = e_t + \sum_{i=1}^{p} \tau_i \bar{z}_{t-i} + \sum_{i=1}^{q} {}_d\psi_i e_{t-i} + \sum_{i=1}^{b} \phi_i u_{t-i} \tag{2.21}$$

Common implementations of time-series analysis in oil and gas include forecasts of oil production in reservoirs [3], global analysis of future production levels in the oil industry [5], forecast of natural gas price [65], oil and gas supply chain [87].

System Identification

In the setting of process control, surrogate models are used to identify input/output relations of dynamical systems. Afterward, the surrogate is typically used for control purposes. In this case, the common time-invariant system framework is used [56], *i.e.*

$$z_t = G_d(q, \tau)u_t + H_d(q, \theta)a_t \tag{2.22}$$

where z, u and e are the output, input and disturbances, respectively, G_d represents the process model, H_d relates the output to disturbances, and β are the model parameters.

Equation 2.22 represents the general structure of the same models described in the previous section by Equations 2.19, 2.20 and 2.21 among others, *i.e.*

- Autoregressive (ARX) models

$$G_d(q,\beta) = \frac{B(q)}{A_d(q)}, \quad H_d(q,\beta) = \frac{1}{A(q)} \tag{2.23}$$

by setting,

$$\beta_p = [\tau_1, \tau_2, ..., \tau_p \quad \phi_1, \phi_2, ...\phi_b] \tag{2.24}$$

and using the operators,

$$A_d(q) = 1 + \tau_1 q^{-1} + ... + \tau_p q^{-p} \tag{2.25}$$

and

$$B(q) = 1 + \phi_1 q^{-1} + ... + \phi_b q^{-b} \tag{2.26}$$

By putting Equations 2.22, 2.28, 2.24, 2.25 and 2.26, we get exactly, Equation 2.19.

- Moving average (MAX) models

$$G_d(q,\beta) = \frac{B(q)}{1}, \quad H_d(q,\beta) = \frac{C(q)}{1} \tag{2.27}$$

- Mixed autoregressive-moving average (ARMAX) models

$$G_d(q,\beta) = \frac{B(q)}{A_d(q)}, \quad H_d(q,\beta) = \frac{C(q)}{A_d(q)} \tag{2.28}$$

System identification techniques haven been successfully applied to oil recovery processes such as, SAGD [107], [112], waterflooding [46], [70], [115] and gas production processes [42].

Dynamic surrogate models

Alternatively, there have been some attempts to identify dynamic surrogate models, which can provide fast approximations of time series outputs of interest as a result of a time series input. For long-term prediction of time-series one of the most common option [7], [43], [95] is to make recursive predictions using a one-step forecast model of the process. This implies, using predicted values at time t, as inputs to predict for time $t + 1$. This procedure, although intuitively, exhibits an error propagation problem, in which, the prediction error at time t is propagated to the subsequent prediction steps. In other words, the error is expected to grow as the number of forecast steps increases. For oil recovery optimization, [32] implemented this approach to find the optimal bottom hole pressure of producing wells at each time step that maximizes net present value after 3600 days.

In the context of time series prediction two (2) strategies have been proposed to mitigate the cited error propagation problem. The first one, known as using Data as Demostrator – DaD [108] attempts to use residual data based on the errors exhibited by the one-step forecast model (base model) and then retraining model. Another option is to perturb the initial data set at each step of the forecasting process to handle more properly the approximated values in the prediction process – RECNOISY [101].

In this work, the conventional SMO framework is extended, and a Dynamic Surrogate Modeling and Optimization (DSMO) approach is proposed. In particular, **Chapter 5** evaluates the recursive approach cited above and discusses its shortcomings and **Chapter 6** formally presents the DSMO approach; in particular, it presents a solution to the time propagation problem.

Chapter 3

Action-value function RL for SAGD optimization[1]

3.1 Introduction

In this chapter, a *tabula rasa* approach is used to solve Equation 2.1, *i.e.* find the optimal steam injection policy for a SAGD process considering a one well pair configuration. This approach consists of using only interactions with the environment or experience, without any supervision or use of human data. Additionally a secondary objective is not only to find an optimal policy but also, to gain insight in the some of the processes cited above. The on-line on-policy SARSA algorithm is used to identify the an approximation of the state-action value function. The environment is a history-matched reservoir numerical simulation model built using data from a SAGD reservoir located in northern Alberta, Canada, the reward function is cumulative net present value considering a discrete action space and continuous state space.

The key findings of this work include:

- An optimal policy should aim to expand the steam chamber until it reaches the overburden and keep a high reservoir temperature throughout the production

[1]A version of this chapter has been published in 2021 in the *Journal of Petroleum Science and Engineering*, 206, 108735

horizon

- Pressure plays a key role until the steam chamber reaches the overburden, afterwards temperature is the driving mechanism of oil production
- Optimal policies seem to exhibit a general shape, characterized by three regions: (1) reach a constant rate, (2) sharp increase and immediately an abrupt decrease until it reaches a minimum value, (3) constant minimum value.

3.2 SARSA with function approximation

In this work a value-based strategy is used, where during the interaction process between the agent and the environment, an action-value function is identified. This function offers the expected return starting from state s, taking action a and then following policy π,

$$q_\pi(s,a) = \mathbb{E}_\pi[G_t \,|\, S_t = s, A_t = a] \tag{3.1}$$

Using the Bellman Expectation Equation this function can be composed into an immediate reward plus the discounted value of the successor state, *i.e.*

$$q_\pi(s,a) = \mathbb{E}_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) \,|\, S_t = s, A_t = a] \tag{3.2}$$

Equations (3.1) and (3.2) represent the general framework of RL where a stochastic MDP is modeled; however in this work we are assuming a deterministic process. The general idea is to find $Q_\pi(S,A)$ which represents an approximation of $q_\pi(s,a)$ only by continuous interactions with the environment[2].

This raises two (2) issues:

[2]Notation-wise, an upper-case letter representing a state or action indicates a specific state (or action) while a lower-case letter corresponds to a generic state (or action)

- The term $q_\pi(S_{t+1}, A_{t+1})$ found in Equation (3.1) represents the current estimate of the state-action value function. This concept is known as Temporal Difference (TD) learning.

- We must use some *update rule* for $q_\pi(s, a)$. Every time the agent encounter state s and takes action a, the value of $q_\pi(s, a)$ must move towards the true value.

A popular used *update rule* is SARSA, i.e.,

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)] \tag{3.3}$$

where α represents the learning rate. If t is the terminal state T then $Q(S_{t+1}, A_{t+1})$ is zero. Equation 3.3 can be seen as an update to an estimated state-value function that shifts its value at a state-pair (S_t, A_t) towards an *update target* for that state. In this case, the term $R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})$ represents the update target and the complete term $R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)$ is also known as the TD error.

SARSA is considered an on-policy method in which $Q(S, A)$ is learned using data generated following the current policy π. More specifically, notice in Equation 3.3 the update is done using $Q(S_{t+1}, A_{t+1})$ where, A_{t+1} is chosen according to the current policy π. Off-policy methods, such as Q-learning, $Q(S, A)$ is learned using data generated following different policies, a random policy for example. Additionally, these algorithms rely only on interactions with the environment for the update of the action-value function, i.e., no domain knowledge is needed. However, it is important to have a domain knowledge for realistic parameter selection and result interpretation.

3.2.1 ϵ-greedy policy

Policy π is pre-defined and as the interaction process continues, it is improved. In order to get a the real state-action value function $q_\pi(s,a)$, the agent must have the capacity to act as best as it knows (current approximation of $q_\pi(s,a)$ and also allow for exploration of new state-action pairs. This is known as the exploration-exploitation trade off and the most commonly used strategy is for the agent to follow a ϵ-greedy policy.

At every time step, a new action is chosen by following a policy improvement mechanism, such as ϵ-greedy policy. In this policy, the agent will choose the optimal action (one that will offer the most return), also known as greedy action with a probability of $1-\epsilon_g$, and a random action with a probability of ϵ_g. Mathematically this can be expressed as,

$$\pi(a|s) = \begin{cases} \frac{\epsilon_g}{m} + 1 - \epsilon_g & \underset{a}{\operatorname{argmax}}\, q_\pi(s,a) \\ \frac{\epsilon_g}{m} & \text{for the rest of the possible actions} \end{cases}$$

3.2.2 Function approximation

In cases where the number of state-action pairs is too big, the tabular setting (detailed above) is not feasible, not just because of the memory needed for large table, but the time and data needed to fill them accurately may be prohibitive. In these cases almost every state-action pair visited will never have been seen before and most likely will never be seen again in the future. Thus it is necessary to generalize between states by parametrizing the value-function, i.e.,

$$\hat{q_\pi}(s,a,\mathbf{w}) = q_\pi(s,a) \tag{3.4}$$

Applying the concept of *update target* mentioned before, we can refer to this update as $s, a \mapsto u$, where s, a is the state-action pair and u is the target that s, a's estimate value is shifted toward. In the case of SARSA (using TD), this will be:

$$S_t, A_t \mapsto R_{t+1} + \gamma Q_\pi(S_{t+1}, A_{t+1}) \tag{3.5}$$

This input-output behavior can be modeled using machine learning methods called supervised learning. In particular when the outputs are numbers this process is referred to as function approximation. Although we could in theory use any supervised learning strategy (e.g., neural networks, support vector machines), not all of them are suited for use in reinforcement learning. The reasons are that in RL:

- data is temporally correlated
- is usually done in an online setting where the full dataset is not available from the beginning
- when using TD, the target label change, for example, in Equation 3.5 because the term $\gamma Q_\pi(S_{t+1}, A_{t+1})$ represents the current estimation, it will change over time

To overcome these issues, methods based on stochastic gradient descend (SGD) are used together with strategies such as, neural networks, coarse coding and polynomials. In SGD, the weights are adjusted after each sample by a small amount in the direction that would reduce the error on that sample. For more detail the reader is recommended Chapter 9 in [99]. The final SARSA algorithm using function approximation is described below.

3.3 Case study

The elements of the reinforcement learning framework are summarized in Table 3.1 and detailed in this section.

Algorithm 2: SARSA with function approximation

```
Initialize $\hat{q_\pi}(s, a, \mathbf{w})$
**for** *each episode* **do**
    Initialize environment
    Choose $a$ from $s$ using policy $\pi$ derived from $\hat{q_\pi}$
    **for** *each time step* **do**
        Take action $a$ and observe $r$ and $s'$
        Choose $a'$ from $s'$ using policy $\pi$ derived from $\hat{q_\pi}$
        Calculate $r + \gamma \hat{q_\pi}(s', a', \mathbf{w})$
        Update $\mathbf{w}$ using step above as output and $(s, a)$ as input
        $s = s', a = a'$
    **end**
**end**
```

RL Element	**Description**
Environment	One well pair numerical reservoir simulation model
State	$S_t = [Np_t \,, Wi_t \,, Wp_t]$
Actions	w.r.t. to steam injection rate in previous time step 1)increase/2)decrease by a constant value and 3) no change
Reward function	Net present value
Transition between states	System of partial differential equations (mass and energy balance)
Policy	$\epsilon_g -$greedy
Function approximation strategy	Stochastic gradient regression
Implementation	Python 3.6 and STARS-CMG 2018

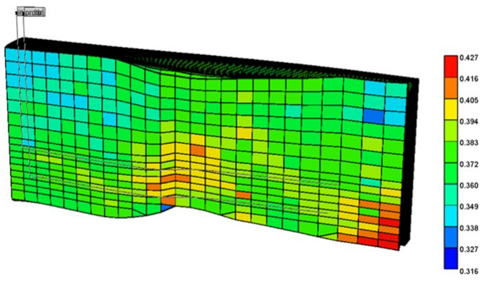
Table 3.1: Summarized description of RL elements

3.3.1 Environment

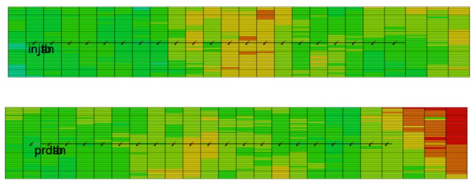
The *environment* is represented by a numerical reservoir simulation model (Figure 3.1) built using data from a SAGD reservoir located in northern Alberta, Canada. The model was history-matched with 1355 days of steam injection/oil production data which consists of three periods, steam circulated from tubing to annulus in both injector and producer (165 days), injector and producer are shut (364 days) and normal SAGD operations (826 days). After this the RL agent begins its interaction with the environment for a production horizon of 250 days. For longer production horizon, e.g., 5 years, the algorithm can also be applied to, with very little changes, however, due to the computational expense, we have shown the proof of concept for 250 days. Longer production horizons and multi-pad reservoir scale optimization can be considered as a next level of study. Note that the general reservoir simulation framework fits very well with the MDP form described in the preceding section. In the context of reservoir simulation we can say for example that the pressure and saturation distribution of time t along with well constraints, is sufficient to describe the pressure and saturation distribution for time $t + 1$.

General reservoir and grid data, production parameters and other properties are specified in Tables 3.2, 3.3, 3.4 and Figures 3.3 and 3.2. Specifically, the grid is composed of 25×50×16 blocks in the x, y and z directions, respectively. Although the considered course grid will have an impact in production performance, the optimization procedure used in this work would not be affected. The reservoir has an initial pressure of 650 kPa, an initial oil and water saturation of 0.8 and 0.2, respectively, and an initial temperature of 16°C. In this work, only a unique set of porosity, permeability and relative permeability are considered, although it is possible to include uncertainty as a part of the RL framework, that issue is left for consideration of future work.

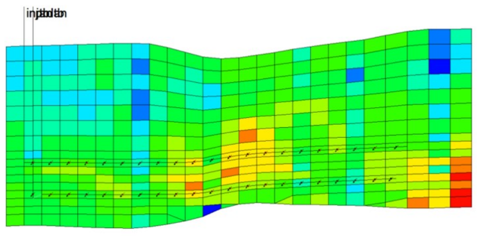
It is worth mentioning that even though, real production data for this reservoir,



(a)



(b)



(c)

Figure 3.1: Porosity field in a)3D view , b) IJ view, plane 9-top and 13-bottom, c) IK view, plane 26 (case study)

only static data was used in this work due to the on-policy nature of SARSA. More specifically, as was described in Section 3.2, SARSA updates it state-action value function $Q(S, A)$ using data generated following the current policy π, thus, using the available production data would not be possible since this data was generated using a different policy.

3.3.2 State

The *state* of the reservoir model must comply with the MDP assumption, i.e., it must carry enough information to capture the history of the process or the previously applied injection rates. In this work the state S is defined as:

$$S_t = [Np_t \, , Wi_t \, , Wp_t] \tag{3.6}$$

Table 3.2: Reservoir simulation model parameters (case study)

Reservoir data	Reservoir compressibility [kPa^{-1}]	7.00×10^{-6}
	Rock heat capacity [$\text{J/m}^3{}^\circ\text{C}$]	2 390 000
	Rock thermal conductivity [J/m day°C]	6.6×10^5
	Reservoir initial temperature [°C]	16
	Reservoir initial pressure [kPa]	650
	Initial water saturation	0.2
	Initial oil saturation	0.8
	Average porosity	0.38
	Average horizontal permeability [mD]	3753
	Average vertical permeability [mD]	2814
Production parameters	Production pressure [kPa]	2667.3
Oil properties	°API	7.62
Production horizon/episode length [d]		250

Table 3.3: Reservoir and grid data (case study)

	x	y	z
Number of gridblocks	25	50	16
Gridblocks size [m]	25	2	1.5
Reservoir size	625	100	24

Table 3.4: Component definition (case study)

	H_2O (aqueous)	Bitumen (oleic)
Molecular weight [kg/gmol]	18.02	534
Critical pressure [kPa]	21997	638.40
Critical temperature [°C]	373.90	780.14
Mass density [kg/m^3]	997	947.25
Liquid compressibility [kPa^{-1}]	0	1.79×10^{-6}
Thermal expansion coefficient (1st coeff) [$^\circ\text{C}^{-1}$]	0	4.51×10^{-4}

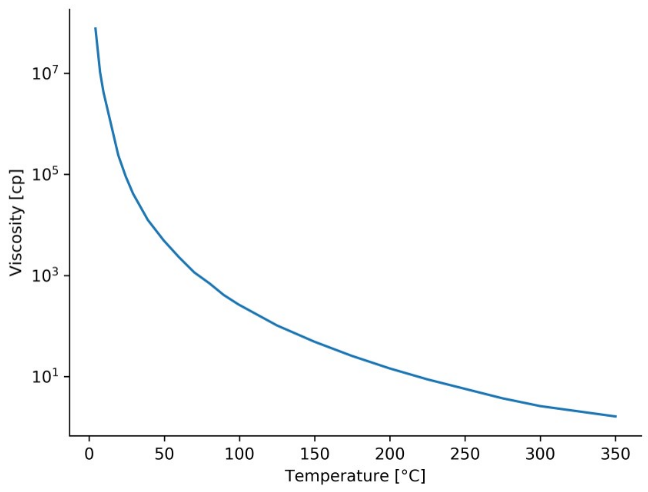


Figure 3.2: Behaviour of viscosity with temperature (case study)

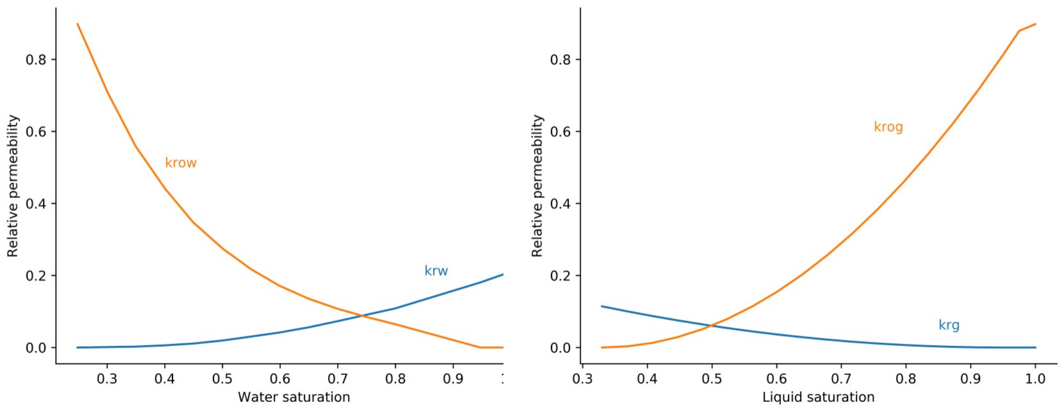


Figure 3.3: Relative permeability curves (case study)

where, Np_t, Wi_t and Wp_t represent cumulative oil production, steam injection and water production, respectively, all at time step t.

This definition relies on the using cumulative values that can implicitly capture the dynamic change in the reservoir without explicitly using time as a variable; furthermore it's validity is proven by the positive results obtained as shown in Section 3.4. However, because we are using a numerical reservoir simulator we have access to different variables, so other valid variables may include, pressure and saturation profiles, steam injection rates at the previous time step, production rates, etc.

3.3.3 Actions

The well has three (3) possible *actions*: increase/decrease steam injection rate of the previous time step by adding a constant value (Δ) and no change. This action space is designed to prevent the steam injection rates from on time step to another from changing dramatically, thus complying with typical SAGD operations constraints. In this work, two separate values of Δ are studied, i.e., $\Delta = 10$ (Case 1) and $\Delta = 5$ (Case 2).

3.3.4 Reward function

The *reward* function corresponds to the classical net present value function for a given time step t, i.e.,

$$R_t = \mathrm{NPV}_t = \frac{P_o q_o - C_{steam} q_s - C_{water} q_w}{1 + i^{\frac{t-t_{ref}}{365}}} \tag{3.7}$$

where, P_o is the oil price [USD/STB], q_o the oil production rate [STB/day], C_{steam} the cost of steam generation [USD/STB], q_s the steam injection rate [STB/day], C_{water} the cost of produced water handling [USD/STB], q_w the water production rate, i is the annual discount factor [fraction], t is the current time and t_{ref} is the reference time to which NPV is discounted. The value of these economical parameters are shown in Table 3.5. The NPV expression (Equation 3.7) corresponds to the reward (R_t) that results of taking a specific action at a specific time step. In the long run, we are interested in maximizing the sum of all NPV values calculated at each time step as stated in Equation 2.1 as J and in Equation 2.16 as G_t.

Table 3.5: Economical parameters (case study)

Parameter	Value
P_o [USD/STB]	50
C_{steam} [USD/STB]	10
C_{water} [USD/STB]	5
i [fraction]	0.1
t_{ref} [day]	0

3.3.5 Transition between states

The *transition between states* represents the dynamic of the world, i.e., for a given state and action, the environment should offer a new state (S_{t+1}) and a reward (R_{t+1}). In SAGD, the dynamic of world corresponds to the flow in porous media phenomena described by a system of partial differential equations (PDEs). The reader is reminded that the goal of RL is not to approximate this function, it is rather considered a black box with which the agents only interacts by taking actions at every time step, transitioning to a new state and receiving a reward.

In particular the set of PDE's are a result of applying mass and energy balance of each of the n_c hydrocarbon components and water in each grid block [82]. This results in a system of $n_{gb}(2n_c + 4)$ equations with the same number of unknowns where $n_g b$ represent the number of grid blocks. These variables correspond to the state variables present in Equation 2.1 and are solved for each time step t in the well-knwon reservoir simulation software CMG-STARS 2018 [19] using the Adaptive-Implicit Method. In this work no chemical reactions were used.

Additionally, two (2) boundary conditions are used, pressure is specified at the producer well as bottom hole flowing pressure (*Dirichlet condition*) and no-flux zones and steam injection rates in the injector well (*Neumann condition*). Regarding initial conditions, the initial saturation field is determined from the depth of the water-oil and gas-oil contacts; the initial pressure field is estimated from pressure at a reference depth (datum), then the pressure in the rest of the grid is calculated using the condition of hydro-static equilibrium.

3.3.6 Policy

An ϵ-greedy policy that decays with time given by,

$$\epsilon_g = \epsilon_g \cdot \epsilon_{decay}^{\text{episode number}} \tag{3.8}$$

where ϵ_{decay} is a decay function parameter set to 0.8.

3.3.7 Function approximation strategy

The *function approximation* strategy was a stochastic gradient regression in which a linear model of the form $f(x) = \mathbf{w}^T \cdot x$ is fitted by minimizing a regularized training error function. This function consists of a loss function that measures model fit and a regularization term that penalizes model complexity. Mathematically, this function in SGR is represented as,

$$E(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} L(y_i(\mathbf{w}), \hat{y}_i(\mathbf{w})) + \beta R_{eg}(\mathbf{w}) \tag{3.9}$$

where L is a loss function, y_i and $\hat{y}_i$ are given by, $R_{t+1} + \gamma \hat{q}(S_{t+1}, A_{t+1}, \mathbf{w}_t)$ and

$\hat{q}(S_t, A_t, \mathbf{w}_t)$, respectively. R_{eg} is a regularization term and $\beta > 0$ is a non-negative parameter. In this work $\beta = 0.015$. Additionally, the state vector (Equation 3.6) was featurized using six (6) radial basis kernels of the following form,

$$f(S_t) = \exp\left(-\frac{\|S_t - c\|^2}{2\sigma^2}\right) \tag{3.10}$$

where c is the center state and σ the width of the function and is also expressed as, $\gamma = \frac{1}{2\sigma^2}$. The gamma value for each radial basis kernel is 0.25, 0.5, 1.0, 2.0, 5.0, and 10.0.

SGR represents a very convenient function approximation strategy for two main reasons, i) although considered a linear model, because it is a linear combination between the weights $\mathbf{w}$ and the input, it is possible to represent non-linear processes, such as SAGD, ii) it offers the possibility to update the weights incrementally as new data is generated (see Section 3.2.2).

3.3.8 Implementation details

The RL algorithm and its elements were implemented in Python 3.6, this includes, the SARSA algorithm, the connection between the environment and SARSA, and the function approximation strategy - SGR 3.3.7). In particular, scikit-learn libraries were used for the SGR model [76] and the environment, represented by a numerical reservoir simulation model was modeled using CMG-STARS 2018 [19].

3.3.9 Verification

To verify the results obtained by the REINFORCE algorithm, a static optimization study is carried out using the well-known reservoir optimization software CMG-CMOST [18]. Here, the steam injection rate at every time step for each well is assumed as an independent variable, for a total of 500 variables each with a range of $20 - 500\text{m}^3/\text{d}$. In particular, the Particle Swarm Optimization algorithm is used.

Inspired by swarm intelligence, Particle Swarm Optimization (PSO) mimics the behavior of groups of species such as bird flocking, in attempts to find shelter or food. In the search, each individual will attempt to balance their own knowledge (exploitation) and of the flock (exploration).

The method consists of P individuals or particles exploring the search space. Each individual knows their own position and velocity and will adjust to a new position based on, the best position visited by itself and the best position visited by any other member of the flock. This update consists of the addition of three (3) components:

- The previous velocity multiplied by an inertia weight that represents a trade-off between exploration and exploitation
- A cognitive learning factor that denotes how much of the individual's own success will impact the new velocity
- A social learning factor that embodies the attraction toward the flock's success

3.4 Optimization results

In this section, we will examine the results of the optimization or learning process. In particular, we are interested in the learning curve exhibited by the agent, and the optimal steam injection rate policy obtained for both cases.

Figure 3.4 shows the learning curve for both cases, where the dark lines represent the moving average of 10 episodes and the light lines the cumulative net present value for each episode after 250 days. Large variations in the NPV of initial episodes can be observed which is due to poor approximations of action-value function by

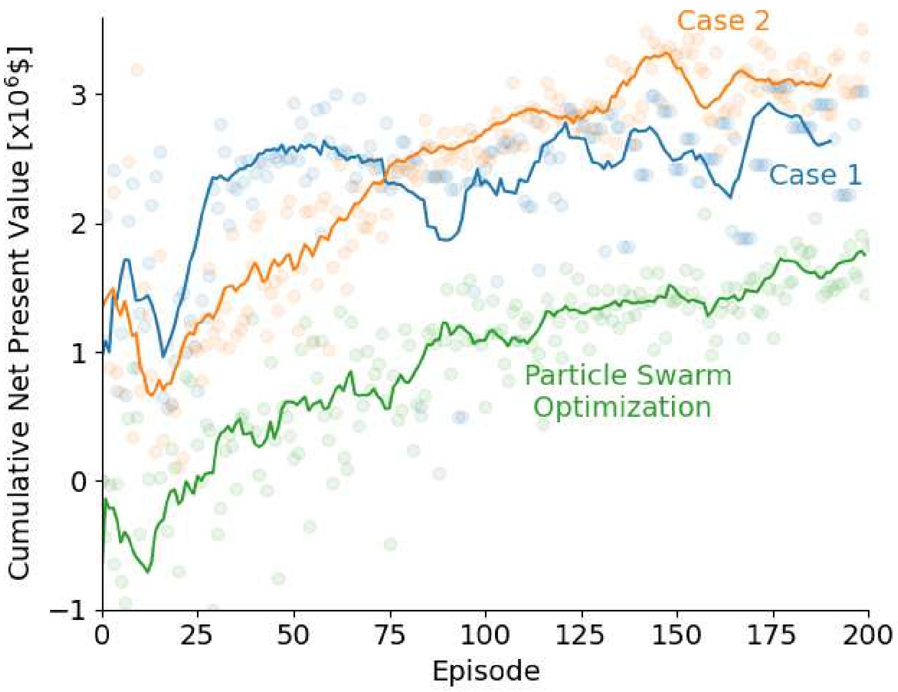


Figure 3.4: Learning curve for Case 1,2 and PSO (CMOST). Each (light-colored) circle represents the cumulative net present value (return G_t) after a production horizon of 250 days for each episode. The dark lines represent a 10-episode moving average.

SGR models and to the fact that the amount of exploration is high (ϵ_g is relatively high, which is a inversely proportional to the number of episodes). As the number of episodes increases, both of these factors are less of an issue, i.e., the SGR models give better estimates and ϵ_g is reduced. In both cases we can also note that after an important improvement of NPV, the agent starts oscillating around a specific value. This phenomenon is know as chattering and has been previously reported to happen using SARSA [33], [34]. Due to the randomness of key parameters (e.g., mainly ϵ_g and to a lesser extent - Equation 3.8, c center state in the radial basis kernels - Equation 3.10) it is possible to obtain different results in a different run, however, here we can see a significant improvement of the cumulative net present value and an arguably stable policy.

On one hand, Case 1 ($\Delta = 10\text{m}^3/\text{d}$) reached convergence after almost 40 episodes at an NPV of approximately \$ 2.7×10^6. On the other hand for Case 2 ($\Delta = 5\text{m}^3/\text{d}$), even though it required 140 episodes to reach convergence, NPV was around \$3.1 $\times$ 10^6. After this, in both cases, the agent oscillates around these values for the rest of the 200 episodes. This difference represents an increment of almost 15% of NPV for Case 2 when compared to Case 1, requiring roughly 3.5 times the number of episodes. We can also report that for Case 2, the improvement is much slower but less oscillating due to a smaller value of Δ, at the cost of needing a larger number of episodes to reach convergence. These two cases are compared to a constant injection *base case* of a rate of 195m^3/d that exhibited an NPV of \$2.54 $\times$ 10^6. Thus, Cases 1 and 2, represent an improvement over the base case of 6.3% and 22%, respectively. Furthermore, Figure 3.7 shows the action taken by the agent at each time step during the optimal steam injection policy for Case 1 and Case 2.

These results can also be compared to the results offered by the PSO algorithm implemented in CMOST shown in Figure 3.4. Note how the cumulative NPV obtained here was around \$1.78 $\times$ 10^6; which represents a reduction of 35% and 42.5%

of the results obtained in Case 1 and Case 2, respectively.

The difference between Case 1 and 2, and CMOST optimal in NPV can be explained in cumulative steam injection, and oil and water production (Figure 3.5). Regarding Case 1 and 2, we can see that despite Case 2 having exhibited a higher cumulative water production, it also produced more oil and injected less steam than Case 1. In particular, Case 2 produced 2.405×10^4m^3 vs 2.055×10^4m^3 produced by Case 1, a 17% increase. Regarding cumulative steam injection, Case 2 injected a total of 2.87×10^4m^3 and Case 1, 3.165×10^4m^3, 10% less as compared to Case 1. In terms of cumulative water production, Case 2 produced 3% more than Case 1, 5.08×10^4m^3 vs 5.254×10^4m^3. With respect to the CMOST optimal case, Case 1 and 2, show a similar amount of cumulative produced water, however, the CMOST case exhibits a much higher amount of cumulative steam injection. Furthermore, a lower cumulative produce oil of 1.98×10^4m^3, which represents a reduction of around 18 percent as compared to Case 2.

Figure 3.6 shows the optimal steam injection curves for both cases (episode number 185 for Case 1 and 2) and the CMOST verification case. Regarding Case 1 and 2, despite the significant differences in the number of episodes necessary to achieve convergence and net present value, the injection policy in both cases shows similarities. We can distinguish three specific regions: [i] reach a constant rate, [ii] sharp increase and immediately an abrupt decrease until it reaches a minimum value, [iii] constant minimum value. However, we can also note differences in each region, so for example, in region [i], Case 1 maintains a constant value of 250 m^3/d (initial rate) while Case 2, decreases to 160 m^3/d and then maintains it constant. In region [ii], Case 1 begins its sharp increase at day 58 and increases to a value of 350 m^3/d, in Case 2, the increase occurs later, at day 80, and then increases to a rate of 225 m^3/d. Another difference is the time in which the profiles reached the minimum value, for Case 1, this was at day 107, and for Case 2, this was later on, at day 139.

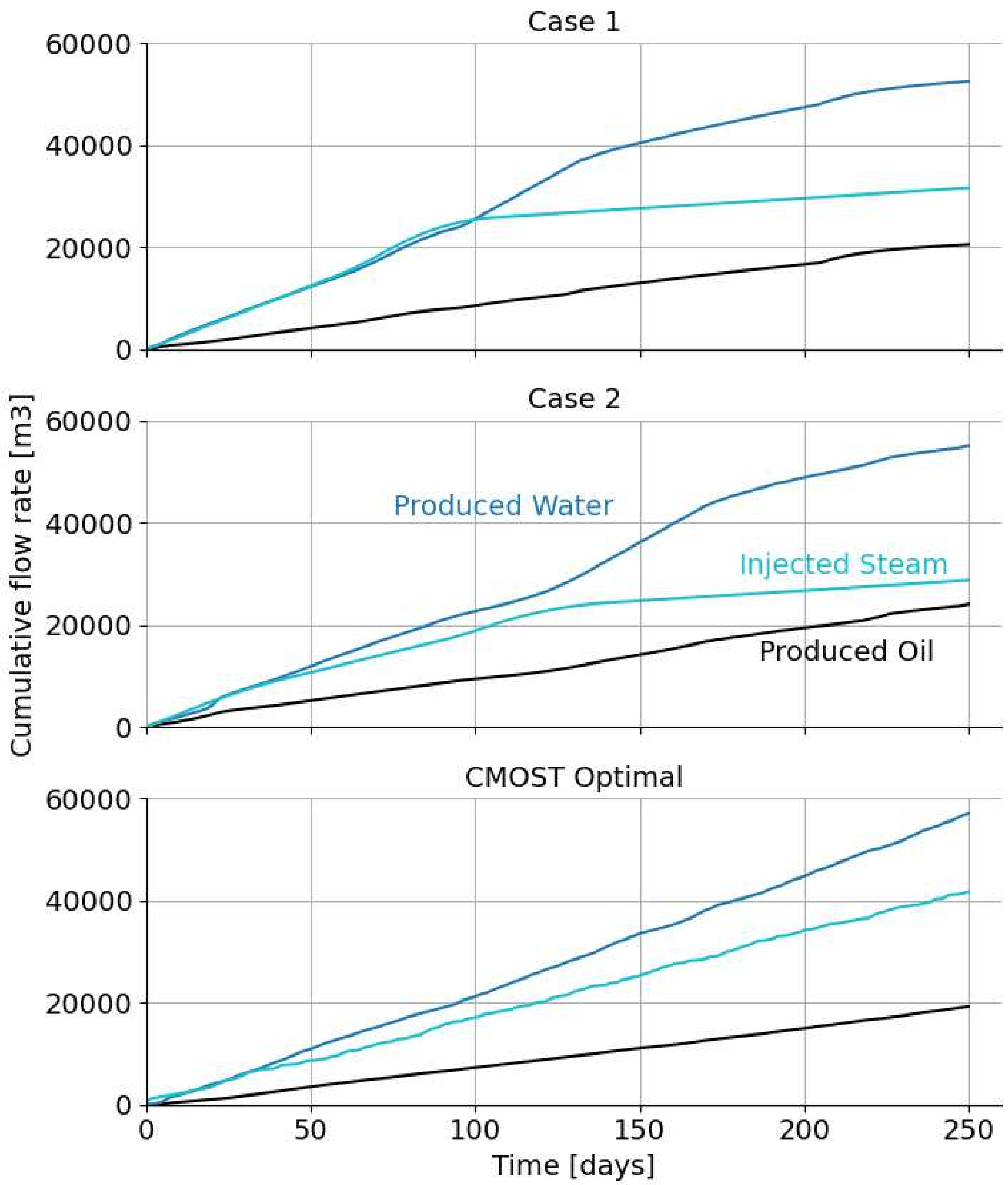


Figure 3.5: Cumulative steam injection and, oil and water production for Case 1 (top) and Case 2 (middle) and CMOST optimal (bottom)

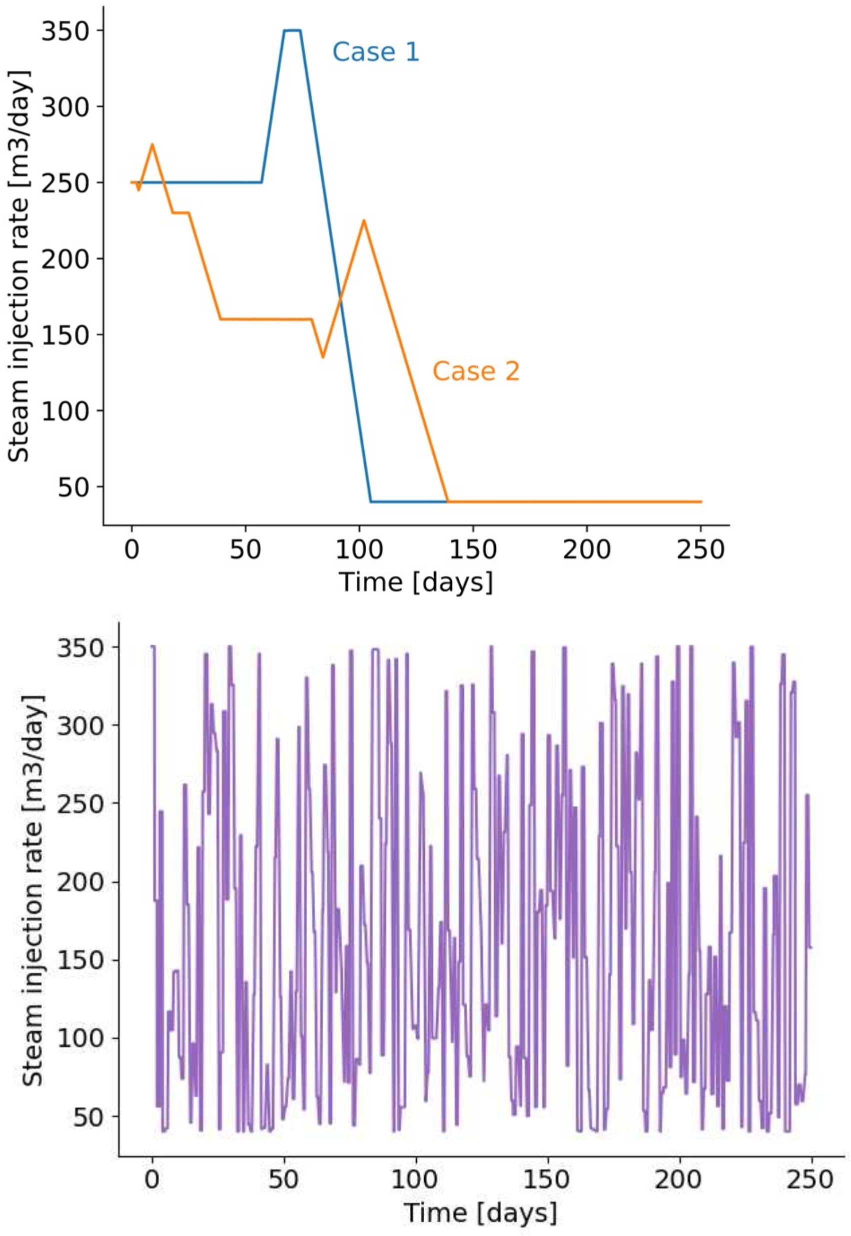


Figure 3.6: Optimal steam injection policy for Case 1 and 2 and CMOST verification case

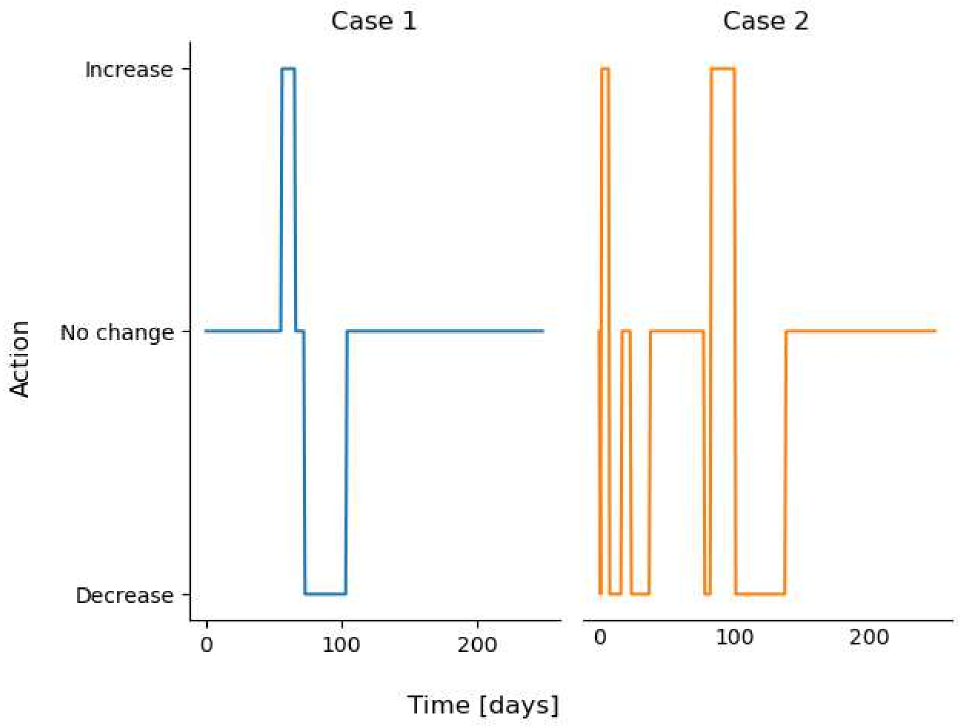


Figure 3.7: Actions taken at each time step by the optimal steam injection policy for Case 1 and Case 2

With respect to the CMOST verification case, we can see very different behavior with respect to Case 1 and 2. There is no clear trend in terms of time, and the steam injection rates vary significantly and drastically throughout the production period.

The training time for Case 1 and 2, was 72.83 and 72.05 hours, respectively, under an Intel Core i7-6700 CPU @ 3.40GHz with 16.0 GB RAM and a 64-bit operating system.

3.5 Physical interpretation of the optimal policy

So far, we have established two things using the cited case study, i.e, an (local) optimum policy was found using RL for different values of Δ - Case 1 and 2 - and, these optimal polices are similar in its general shape, but different in the *parameters* of this shape. These *parameters* are, the constant initial steam rate value, the time in which the sharp increase starts, the value of the rate at which it increases to and the time in which it reaches the minimum value. But, how can we justify these optimal polices from a physical point of view? Why are these policies successful? and more importantly, what can we learn from these policies? In this section, we will answer the first two question and the third is discussed in the following section.

The main aspect that explains the success of the RL-obtained policies is that it is able to achieve and maintain a high temperature even when the steam injection rate is minimum. So for example we can see in Figure 3.8 how the average reservoir temperature increased throughout the production horizon until it reached a stable value of around 32°C (for Case 1) and 34°C (for Case 2). In Figures 3.9 and 3.10 we can also see how the steam chamber grew vertically for roughly the first half of the production horizon in both cases until it reached the overburden. Afterward, there was little change in the temperature within the steam chamber. The time at which the steam chamber reaches the overburden seems to be correlated with the time in which the average reservoir pressure reaches its maximum value. In

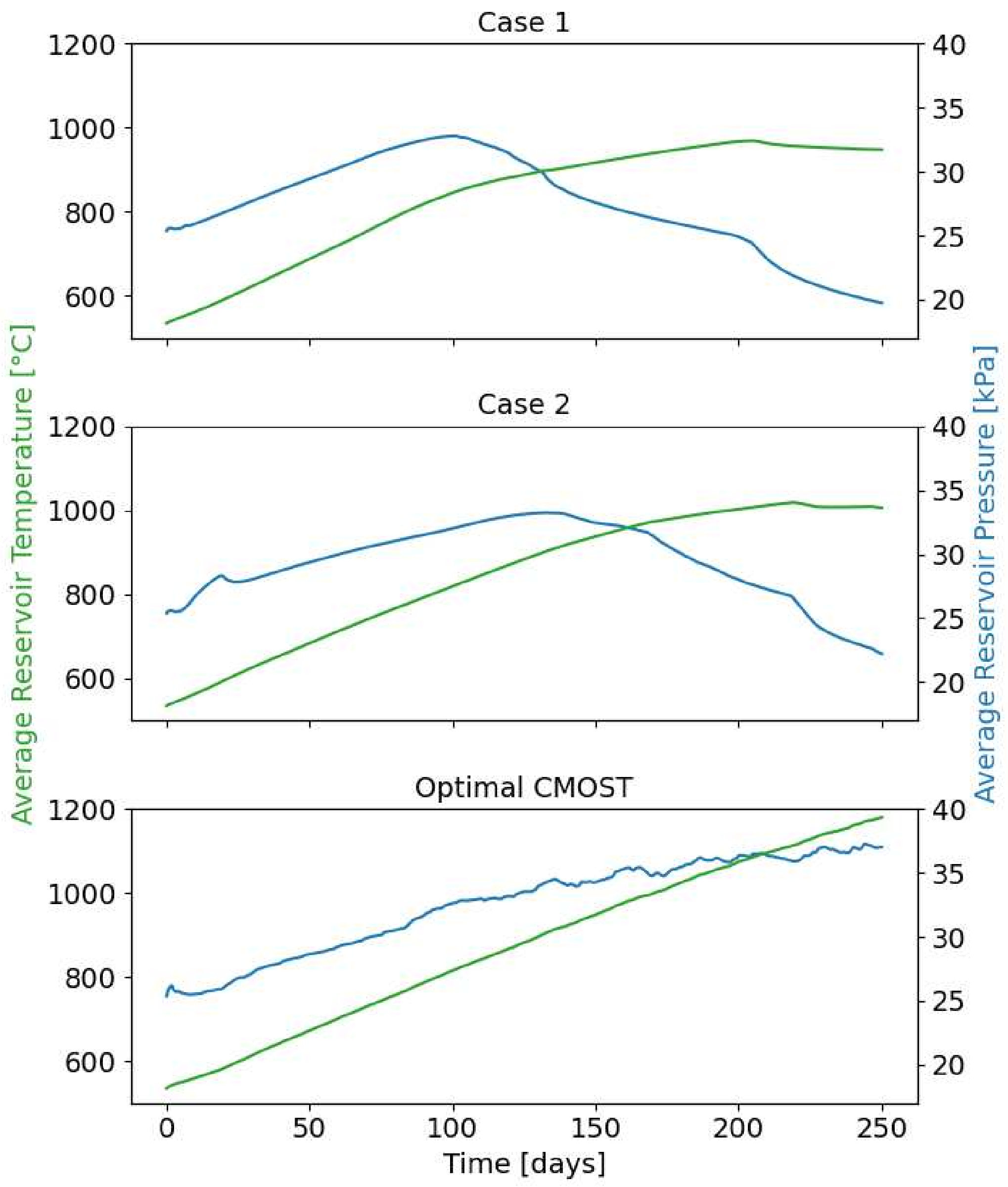


Figure 3.8: Average reservoir temperature and pressure for optimal steam injection policy in Case 1 (top) and Case 2 (middle) and CMOST optimal (bottom)

Figure 3.8 the average reservoir pressure (in blue) increases steadily during the first 107 (for Case 1) and 139 (for Case 2) days and then reduced until the end of the horizon. Interestingly enough, these days also match the time in which the steam injection rate starts to reduce. Case 2 offers a very slight but interesting difference with respect to Case 1 in terms of temperature. From Figure 3.8 we can see that initially, the temperature exhibited by Case 1 was higher than for Case 2, however, afterward, the trend was inverted despite Case 2 having a lower steam injection rate. We can also note this in Figures 3.9 and 3.10 - middle right figures, notice the higher temperature for Case 2. Unlike the behavior exhibited by the RL-obtained policies, the CMOST-obtained policy shows a pressure-increasing tendency towards the end of the production horizon. Similarly, the average reservoir temperature is also increasing, reaching a value of 37 degrees C after 250 days, as opposed to Case 1 and 2, where as noted earlier there was a decrease in temperature until reaching a final value of between 20 and 22 degrees C.

Additionally Figure 3.11 shows the steam injection pressure for optimal policy obtained in Case 1 (left) and Case 2 (right). Note how Case 1 exhibits a higher steam injection pressure due to the higher steam injection rates during the first 100 days, as compared to Case 2. However, we can also note, that after these 100 days, the rate of decrease of steam injection pressure is also greater for Case 1 (with respect to Case 2), due to the more sudden drop of steam injection rates, from 350 m^3/d to 40 m^3/d.

These results suggest that this is caused by the differences in region 2 (sharp increase and immediately an abrupt decrease until it reaches a minimum value) in the policies. More specifically, Case 2, exhibited a later sharp increase than Case 1 and also the value of the rate at which it increases was much less. This apparent minor difference explains the higher average reservoir temperature showed by Case 2 and ultimately its better performance over Case 1.

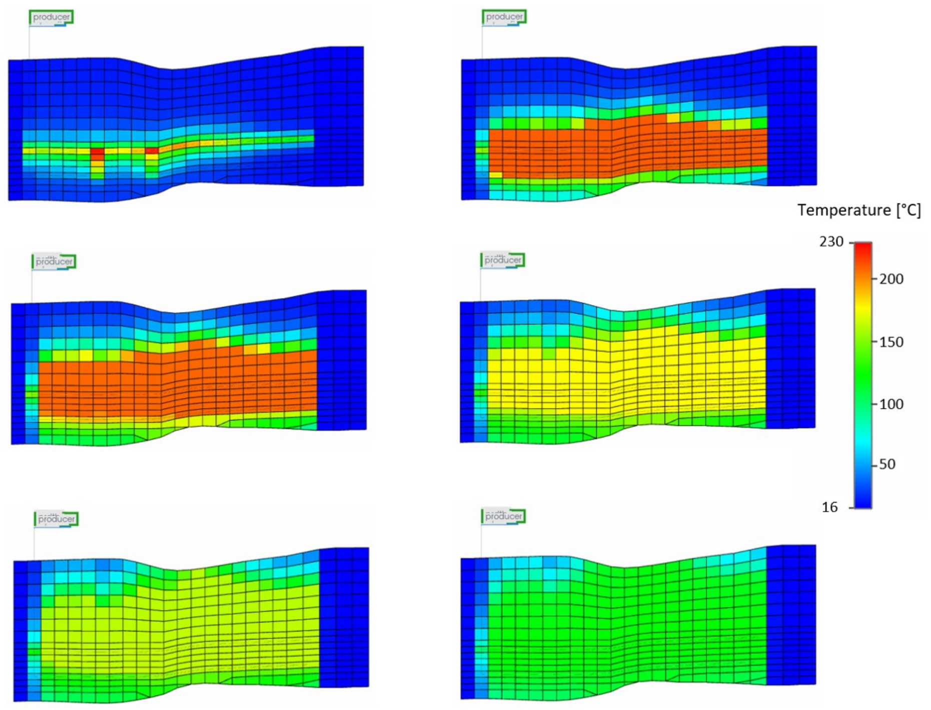


Figure 3.9: 2D Temperature profile in the middle IK plane at time 0 (top-left), 50 (top-right), 100 (middle-left), 150 (middle-right), 200 (bottom-left) and 250 (bottom-right) days for Case 1

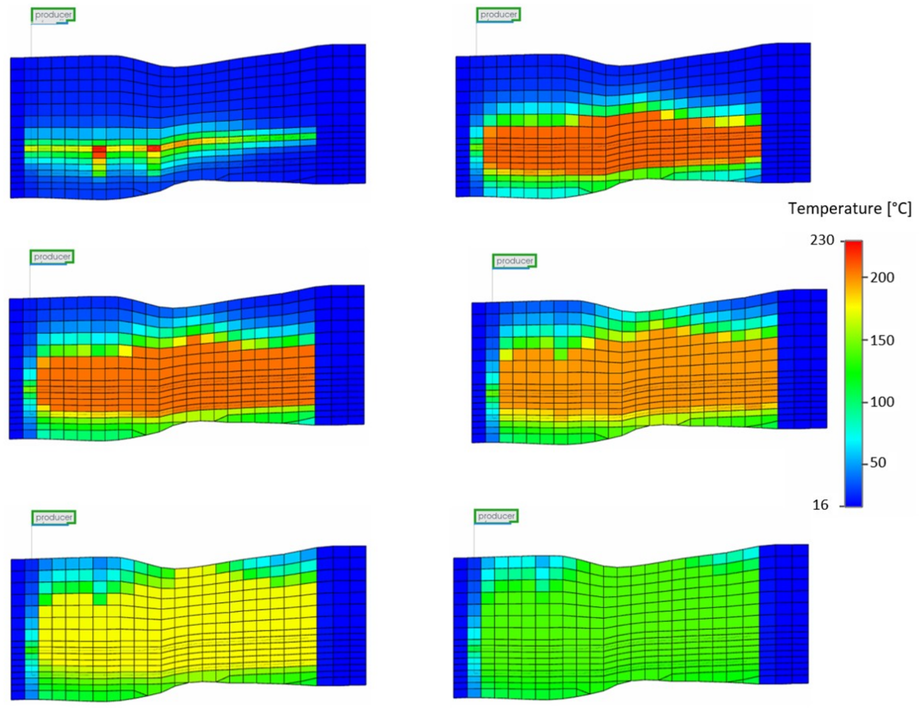


Figure 3.10: 2D Temperature profile in the middle IK plane at time 0 (top-left), 50 (top-right), 100 (middle-left), 150 (middle-right), 200 (bottom-left) and 250 (bottom-right) days for Case 2

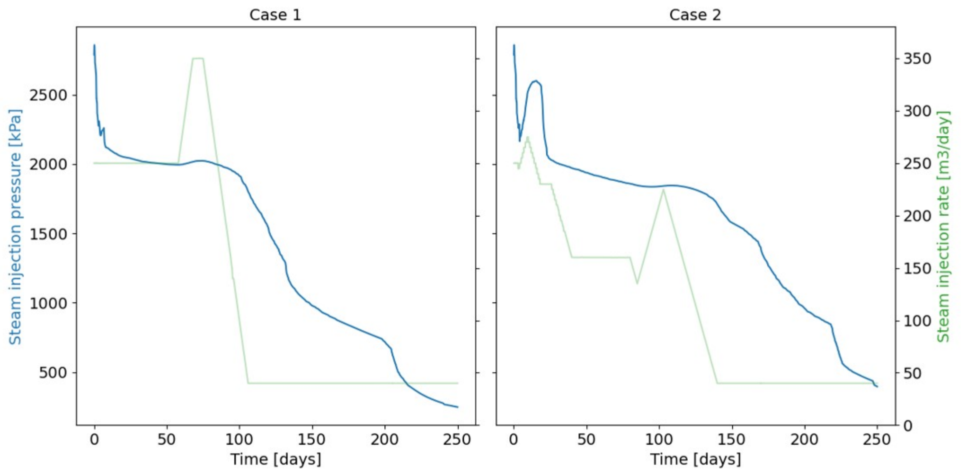


Figure 3.11: Steam injection pressure for optimal steam injection policy in Case 1 (left) and Case 2 (right)

3.6 Summary

This chapter presents the first application of reinforcement learning (RL) approach for the optimization of steam injection rates in a SAGD process. In particular, the objective is to find the steam injection rates at every time step that will maximize overall net present value at the end of the production horizon, also known as the optimal policy.

For the cited case study, we can draw important lessons for SAGD operations, i.e,

- An optimal policy should aim to expand the steam chamber until it reaches the overburden and keep a high reservoir temperature throughout the production horizon
- Pressure plays a key role until the steam chamber reaches the overburden, afterwards temperature is the driving mechanism of oil production

- Optimal policies seem to exhibit the same general shape, characterized by three regions: (1) reach a constant rate, (2) sharp increase and immediately an abrupt decrease until it reaches a minimum value, (3) constant minimum value. However, the *parameters* of this shape may change. We have defined these *parameters* as: the constant initial steam rate value, the time in which the sharp increase starts, the value of the rate at which it increases to and the time in which it reaches the minimum value.
- An initial constant steam injection rate (region 1) is required to increase the average reservoir pressure and allow the steam chamber reach the overburden
- The sharp increase and abrupt decrease in steam injection rate (region 2) appears to have the effect of increasing average reservoir temperature
- In region 3, a minimum steam injection rate is required to maintain the average reservoir temperature

Chapter 4

Policy gradient RL for SAGD optimization

4.1 Introduction

In this chapter, we present the results of implementation of a policy gradient approach to solve Equation 2.1 and find the optimal steam injection strategy of a multi-well SAGD process (environment). The environment is a reservoir simulation model inspired by a reservoir located in northern Alberta, Canada, the action space is discrete and the policy π is parametrized with a deep neural network using the TensorFlow libraries implemented in Python. Additionally, a physics-based interpretation of the optimal policy is presented to contribute to the discussion of identifying key characteristics of optimal policies for SAGD.

Key findings of this work include:

- Optimal steam injection policy 1) an increase or slight increase of steam injection rates, and 2) a sharp decrease until reaching the minimum value
- RL poses as an effective alternative considering that, the mathematical model of the reservoir dynamics is not required, maximization is performed over the entire production horizon

4.2 Reinforcement learning and policy gradient

In policy gradient methods a parametrized policy is learned - $\pi(s|a, \boldsymbol{\theta})$ and the actions are selected by evaluating π without necessarily using a value function. Here, we are interested in maximizing some scalar performance measure $J(\boldsymbol{\theta})$ with respect to the policy parameter $\boldsymbol{\theta}$. Consequently, we want to update the parameters $\boldsymbol{\theta}$ following a stochastic gradient ascent approach, *i.e.*

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \widehat{\nabla J(\boldsymbol{\theta}_t)} \tag{4.1}$$

where $\widehat{\nabla J(\boldsymbol{\theta}_t)}$ represents a sample of the gradient of the performance measure with respect to its argument $\boldsymbol{\theta}_t$. This is only valid provided $\pi(s|a, \boldsymbol{\theta})$ is differentiable with respect to its parameters. There are two main algorithms to solve this problem: REINFORCE and actor-critic methods. In particular, there are two variants of REINFORCE typically used: Monte Carlo and using a baseline. The Monte Carlo variant is described below.

Monte Carlo Policy Gradient

Using the policy gradient theorem [100], we can say:

$$\nabla J(\boldsymbol{\theta}) = \mathbb{E}_\pi \left[G_t \frac{\nabla \pi(A_t|S_t, \boldsymbol{\theta})}{\pi(A_t|S_t, \theta)} \right] \tag{4.2}$$

where A_t and S_t represent the action and state at time t, respectively, G_t is the discounted return, expressed in Equation 2.16. Keeping the literature convention in RL [99] capital letters represent random variables (*e.g.* A_t) whereas their instantiations (*e.g.* a_t) are denoted in lower case.

Using the sample given by Equation 4.2 we can re-write Equation 4.1 as,

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \left[G_t \frac{\nabla \pi(A_t|S_t, \boldsymbol{\theta})}{\pi(A_t|S_t, \theta)} \right] \tag{4.3}$$

This update gives rise to the REINFORCE algorithm [99]. The term inside the bracket represents the update which is given by, the product of the return G_t by the gradient of the probability of taking action A_t divided by the probability of taking that action. This update will favor actions that yield the highest return and encourage taking action that haven't been selected frequently. Also note that REINFORCE represents a Monte Carlo algorithm since it uses the complete return G_t from time t until the end of the episode. The pseudocode corresponding to the REINFORCE algorithm is given below.

Algorithm 3: REINFORCE: Monte-Carlo Policy-Gradient algorithm

```
Input: a differentiable policy parametrization $\pi(a|s, \boldsymbol{\theta})$
Algorithm parameter: step size $\alpha > 0$
Initialize policy parameters $\boldsymbol{\theta}$
for each episode do
    Generate an episode $S_0, A_0, R_1, ..., S_{T-1}, A_{T-1}, R_T$ following $\pi(\cdot|\cdot\boldsymbol{\theta})$
    for each time step do
        $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\gamma^t G \nabla \ln \pi(A_t|S_t, \boldsymbol{\theta})$
    end
end
```

Notice the last line represents a compact version of Equation 4.3 as, $\nabla \ln x = \nabla x / x$.

Action selection

A typical parameterization in cases where the action space is discrete and not large is to define a parameterized numerical preference for each state-action pair $h(s, a, \boldsymbol{\theta})$. This parameterization is arbitrary, and for linear models this would be:

$$h(s, a, \boldsymbol{\theta}) = \boldsymbol{\theta}^T \mathbf{x}(s, a) \tag{4.4}$$

where $\boldsymbol{\theta}$ are the policy parameters and $\mathbf{x}(s, a)$ is the state and action information. For action selection, it is intuitive to have a mechanism that offers high probabilities to the actions with the highest numerical preference. One such mechanism is to select actions according to an exponential soft-max distribution,

$$\pi(a|s, \boldsymbol{\theta}) = \frac{\exp h(s, a, \boldsymbol{\theta})}{\sum_b \exp h(s, a, \boldsymbol{\theta})} \tag{4.5}$$

where the denominator represents the sum of probabilities for each action.

Advantages

There are three relevant advantages of policy gradient methods over action-value for SAGD optimization, i.e.,

- Policy may be a simple function to approximate than action-value
- The choice of parametrization is a good way to include prior knowledge of the learning system
- The approximate policy can approach a deterministic policy according to a soft-max in action preferences

4.3 Case study

In this section, all of the elements of the RL framework (*e.g.* environment, actions, state) are specified in the context of SAGD optimization.

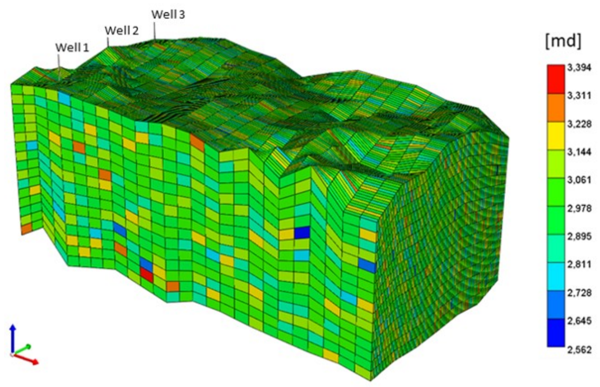


Figure 4.1: 3D grid and porosity field (case study)

4.3.1 Environment

The environment is represented by a three (3) well pairs numerical reservoir simulation model (Figure 4.1) inspired by a reservoir located in northern Alberta, Canada. General reservoir parameters (Table 4.1), grid data (Table 4.2), oil viscosity changes with temperature (Figure 4.2), fluid model (4.3) and relative permeability curves (Figure 4.3) are shown below. Specifically, the grid is composed of 26×93×16 blocks in the x, y and z directions, respectively. The reservoir has an initial pressure of 1075 kPa, an initial oil and water saturation of 0.77 and 0.23, respectively. Additionally, production horizon considered in this case study is 250 days; however, the optimization can be continued for the entire life cycle of the reservoir.

4.3.2 Actions

Two (2) well pairs are considered in this work and each injector well has three (3) possible actions: increase or decrease the steam injection rate from the previous time step by adding/subtracting a preestablished value (Δ) and no change. However, we assume that there is hydraulic communication between the well pairs, so the agent has a total of nine (9) possible actions, corresponding to the combination of each

Table 4.1: General reservoir simulation model parameters (case study)

Reservoir data	Reservoir compressibility [kPa^{-1}]		7.00×10^{-6}
	Rock heat capacity [J/m^3°C]		2 390 000
	Rock thermal conductivity [J/m day°C]		6.6×10^{5}
	Reservoir initial temperature [°C]		16
	Reservoir initial pressure [kPa]		1075
	Initial water saturation		0.23
	Initial oil saturation		0.77
	Average porosity		0.36
	Average horizontal permeability [mD]	Well 1	4000
		Well 2	3000
		Well 3	2000
Production parameters	Production bottom-hole pressure [kPa]		600
Oil properties	°API		7.62
	Thermal expansion [°C^{-1}]		4.51×10^{-4}
	Density [kg/m^3]		947.25
	Compressibility [kPa^{-1}]		1.79×10^{-8}
Production horizon/episode length [d]			250

Table 4.2: Grid data (case study)

Number of gridblocks	x	26
	y	93
	z	16
Average gridblock thickness	1.30	
Pore volume [m^3]	Total	1 463 279
	Average per gridblock	37.82

Table 4.3: Component definition (case study)

	H2O (aqueous)	Bitumen (oleic)
Molecular weight [kg/gmol]	18.02	534
Critical pressure [kPa]	0	638.40
Critical temperature [°C]	0	780.14
Mass density[1] [-]	0	947.25
Liquid compressibility [kPa^{-1}]	0	1.79×10^{-6}
Thermal expansion coefficient [°C^{-1}]	0	4.51×10^{-4}

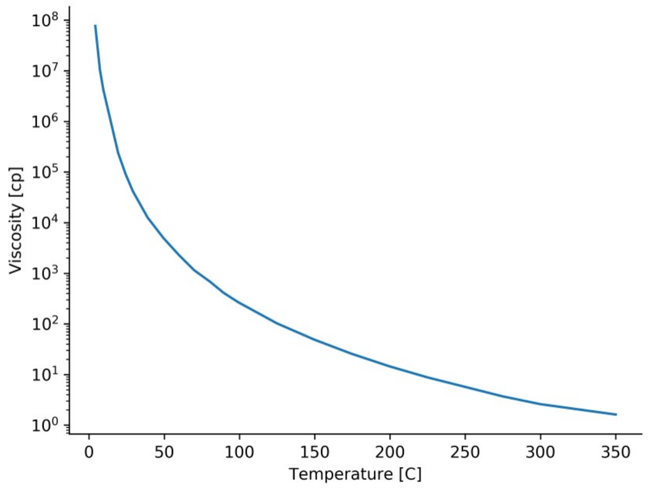


Figure 4.2: Change of viscosity with temperature (case study)

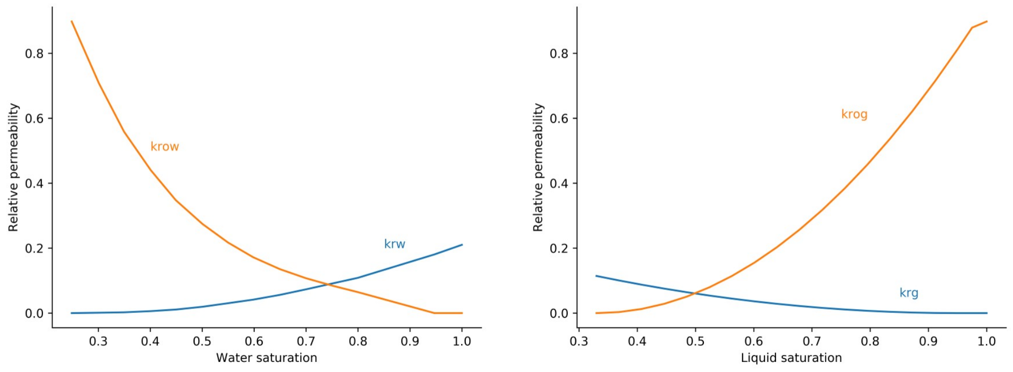


Figure 4.3: Relative permeability curves (case study)

possible action of each well pair. Therefore, for example, one of those nine actions is to, increase the steam injection rate for well 1 and decrease it for well 2.

4.3.3 State

Defined as:

$$S_t = [Np_t \, , Wi_t \, , q_{w,1_{t-1}} \, , q_{w,2_{t-1}}] \tag{4.6}$$

where, Np_t, Wi_t and Wp_t represent cumulative oil production, steam injection and water production, respectively, all at time step t, and q_{w_1} and q_{w_2} are the steam injection rates of well 1 and 2.

4.3.4 Reward function

The *reward* function represent the economic benefits obtained by at time t as a result of taking a specific action at time $t-1$. This is defined as the net present value:

$$R_t = \mathrm{NPV}_t = \frac{P_o \sum_{j=1}^{n} q_{o,j} - C_{steam} \sum_{j=1}^{n} q_{s,j} - C_{water} \sum_{j=1}^{n} q_{w,j}}{1 + i^{\frac{t-t_{ref}}{365}}} \tag{4.7}$$

where, P_o is the oil price [USD/STB], $q_{o,j}$ the cumulative oil production obtained in time step t for well pair j [STB], C_{steam} the cost of steam generation [USD/STB], $q_{s,j}$ the the cumulative steam injection applied in time step t for well pair j [STB], C_{water} the cost of produced water handling [USD/STB], $q_{w,j}$ the water production observed in time step t for well pair j, i is the annual discount factor [fraction], t is the current time and t_{ref} is the reference time to which NPV is discounted. The value of these economical parameters are shown in Table 4.4. The general objective of this work is to maximize the sum of all NPV values calculated at each time step as stated in Equation 2.1 as F and in Equation 2.16 as G_t.

Table 4.4: Economical parameters (case study)

Parameter	Value
P_o [USD/STB]	50
C_{steam} [USD/STB]	10
C_{water} [USD/STB]	0
i [fraction]	0.1
t_{ref} [day]	0

4.3.5 Policy parametrization

A deep neural network (DNN) was used to parametrizeed the policy, *i.e.* $\pi(s|a, \boldsymbol{\theta})$ where $\boldsymbol{\theta}$ represents the weight vector used to make non-linear transformation among the hidden layers. The DNN consists of three (3) hidden layers with 70 neurons per layer; a RELU activation function in each of the hidden layers and softmax for the last layer. Additionally, backpropagation with Adam optimization was used to find

$\boldsymbol{\theta}$ at the end of each episode. The implementation was done in Python using the Tensorflow framework [60].

4.3.6 Transition between states

The *transition between states* represents the dynamic of the world, *i.e.* for a given state and action, the environment should offer a new state (S_{t+1}) and a reward (R_{t+1}). In SAGD, the dynamic of the world corresponds to the flow in porous media phenomena described by a system of partial differential equations (PDEs), i.e., mass conservation, and energy conservation. In this work, this is modeled using a three-phase multi-component thermal, CMG-STARS and an Adapative-Implicit (AIM) solution method. The reader is reminded that the goal of RL is not to approximate this function, it is rather considered a black box with which the agent only interacts by taking actions at every time step, transitioning to a new state, and receiving a reward.

4.3.7 Verification

To verify the results obtained by the REINFORCE algorithm, a static optimization study is carried out using the well-known reservoir optimization software CMG-CMOST [18]. Here, the steam injection rate at every time step for each well is assumed as an independent variable, for a total of 500 variables each with a range of $20 - 500\text{m}^3/\text{d}$.

The biologically inspired algorithms Particle Swarm Optimization (PSO) is used for this purpose. Inspired by swarm intelligence, PSO mimics the behavior of groups of species such as bird flocking, in attempts to find shelter or food. In the search, each individual will attempt to balance their own knowledge (exploitation) and of the flock (exploration).

The method consists of P individuals or particles exploring the search space.

Each individual knows their own position and velocity and will adjust to a new position based on, the best position visited by itself and the best position visited by any other member of the flock. This update consists of the addition of three (3) components:

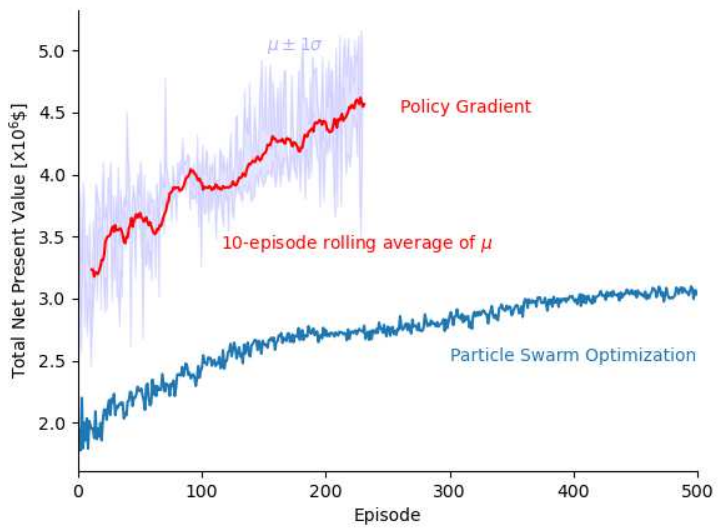
- The previous velocity multiplied by an inertia weight that represents a trade-off between exploration and exploitation
- A cognitive learning factor that denotes how much of the individual's own success will impact the new velocity
- A social learning factor that embodies the attraction toward the flock's success

4.4 Results and Discussion

In this section the results are presented and discussed in two parts: i) the optimization process and the optimal steam injection policy obtained, and ii) the physical interpretation of the optimal steam injection policy.

Optimization results

Figure 4.4 shows the learning curve exhibited by the policy gradient algorithm (red) considering 10 different runs or seeds. Here, the red solid line represents the running average of the mean of the 10 runs, and the light blue line is one (1) standard deviation. Note the improvement of NPV as the agent gained more experience interacting with the environment. In particular, the total NPV value per episode increased from between $3 - 3.5 \times 10^6$\$ to around 4.5×10^6\$, an increase of between 128 to 150 percent. However, We can also note a significant variability throughout the number of episodes.



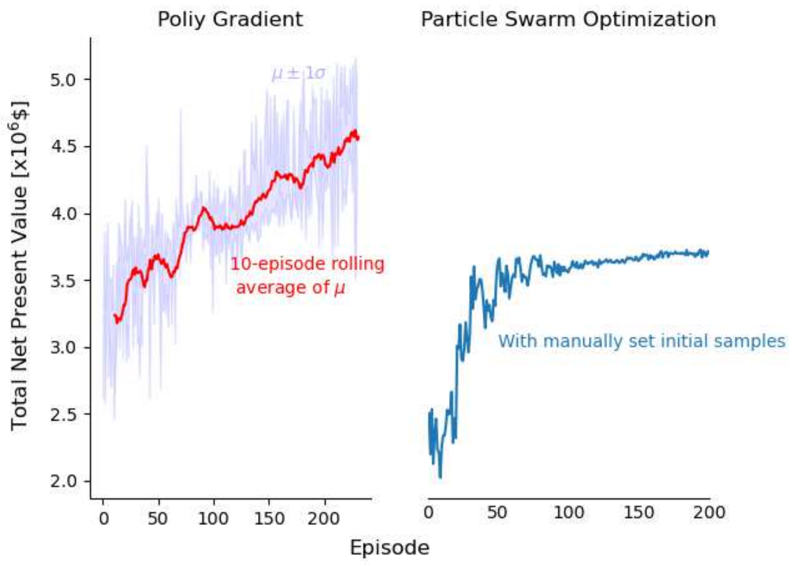


Figure 4.4: Learning curves. The RL-obtained (red) curve considers 10 seeds, the dark red represents the running average of the mean of the 10 runs, and the light blue lines is one (1) standard deviation. In the top figure, the blue curve represents the performance of the PSO algorithm obtained in CMOST using the default experiments generation configuration. In the bottom figure the PSO algorithm was initiated using manually set samples offered by RL

These results can be compared to the learning curve obtained using CMOST (as described in Section 4.3.7) and PSO in the blue curve in Figure 4.4. We offer two cases, the first one (Figure 4.4-top), the default experiments generation configuration for PSO was set and we see that how after 250 episodes, PSO could only achieved an NPV of around 2.75×10^6\$. The second case, shown in Figure 4.4-bottom, considers manually set initial samples for the PSO algorithm. In particular, the first five samples given by the policy gradient algorithm were hard-coded in the PSO initialization framework. Here we see that after 200 episodes, PSO achieved an NPV value of around 3.75×10^6\$

From Figure 4.4 we have selected a typical high-performing episode as the optimal case given by policy gradient, exhibiting an NPV value of ie 4.56×10^6\$, which represents an average value. This optimal steam injection policy is shown in Figure 4.5-top. Notice how, even though the curve for each well is different, the follow a similar shape. More specifically, in both cases, we can observe two distinct regions: 1) an increase (Well 1) or slight increase (Well 2) of the steam injection rate, and 2) a sharp decrease until reaching the minimum value of $20\text{m}^3/\text{d}$. The main difference appears in region 1, for Well 1, we can observe a sustainable increase of the steam injection rate until reaching a maximum value of $240\text{m}^3/\text{d}$, while for Well 2, there is a slight increase to around $75\text{m}^3/\text{d}$ and remains approximately constant for around 100 days. Despite this difference, in both wells, the steam injection rate suffers a sharp decrease (Region 2) at the same point (t_{drop}), i.e., 100 days.

This important difference between the steam injection policy exhibited by well 1 and well 2 can be explained by the heterogeneity of the reservoir. As noted in Table 4.1, the average permeability of the reservoir simulation model for well 1 and 2 are considerable different, 4000 and 3000 mD, respectively. As a consequence, less steam injection is needed for the less permeable region of the reservoir.

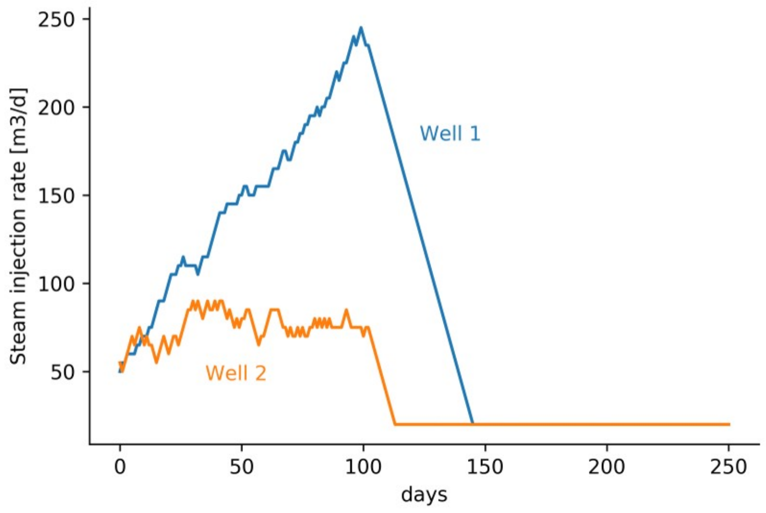
Regarding the CMOST-obtained (4.5-bottom) optimal policy, using the manually set initial samples, we see a radically different behavior. This policy exhibits a very erratic behavior with high changing rates for both wells between the set bounds for steam injection rate values.

4.4.1 Physical interpretation of the optimal policy

Due to the clear tendency is shown in the RL-obtained optimal policy (as opposed to the CMOST-obtained optimal policy) we can further analyze the daily NPV curve (Figure 4.6) to can gain more insight on the value of t_{drop}. Initially, the reward values follow a downhill trend, decreasing from an initial value of 52,534$ to 8.180$ precisely at t_{drop} =100 days. At this point, the amount of reward collected by the agent increases sharply to a value of 36.483$, then begins a slow decrease and then at around 200 days, a sudden decrease until reaching negative values of around -2.800$ for the rest of the episode.

Considering the economic parameters considered (Table 4.4), this behavior is due to the oil production and steam injection rates shown in Figure 4.7-top. Note how, for both wells, initially oil production rates are between 45 and 60m^3/d; in the case of Well 1 this carries on until day 150 and for Well 2 it remains until day 230. So despite the steam injection rate dropping abruptly at t_{drop} =100 days, oil production rates are only affected after at least 50 days for Well 1 and 130 days for Well 2. In fact, in Well 2, we can note even a slight increase in its oil production rate right after t_{drop}. Another interesting point is that even though the steam injection rate values for Well 1 is considerably lower than Well 2, the oil production rates in both cases, are within the same range. The CMOST-obtained optimal policy (Figure 4.7-bottom) shows a different picture, here we see rates oscillating from 40 to 80m^3/d with no particular trend.

The average reservoir pressure and temperature data for the RL-obtained optimal policy (Figure 4.8-top) suggests that the agent's objective during Region 1, is to sus-



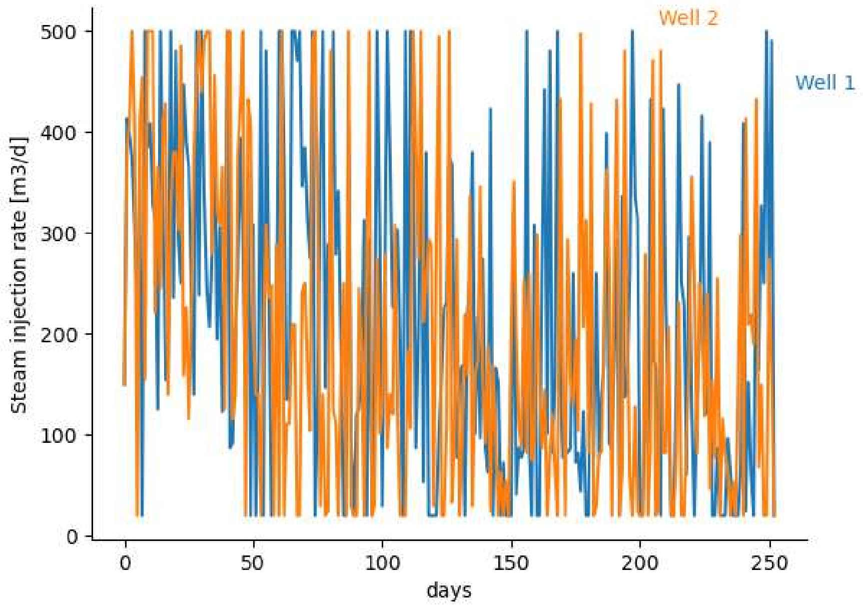


Figure 4.5: RL-obtained (top) and CMOST-obtained (bottom) optimal steam injection policy for both well pairs

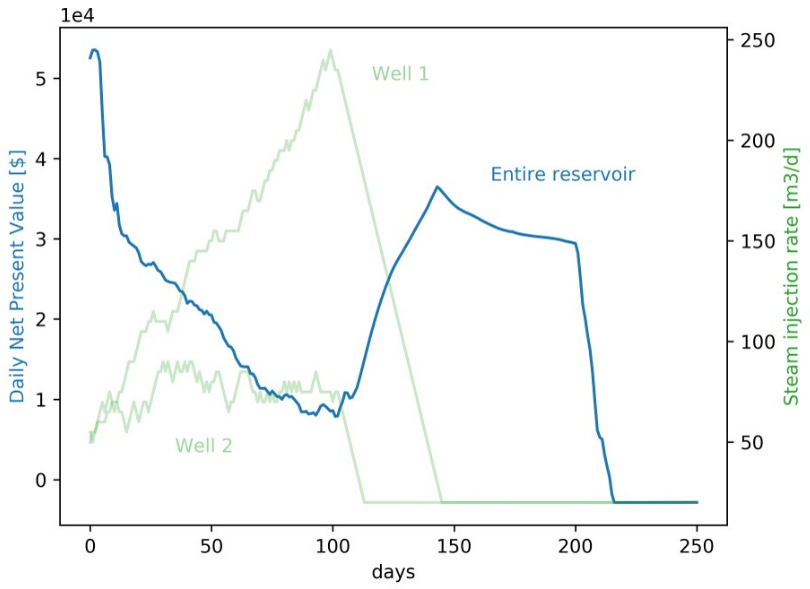


Figure 4.6: Daily Net Present Value of the entire reservoir and steam injection rates in the RL-obtained optimal case

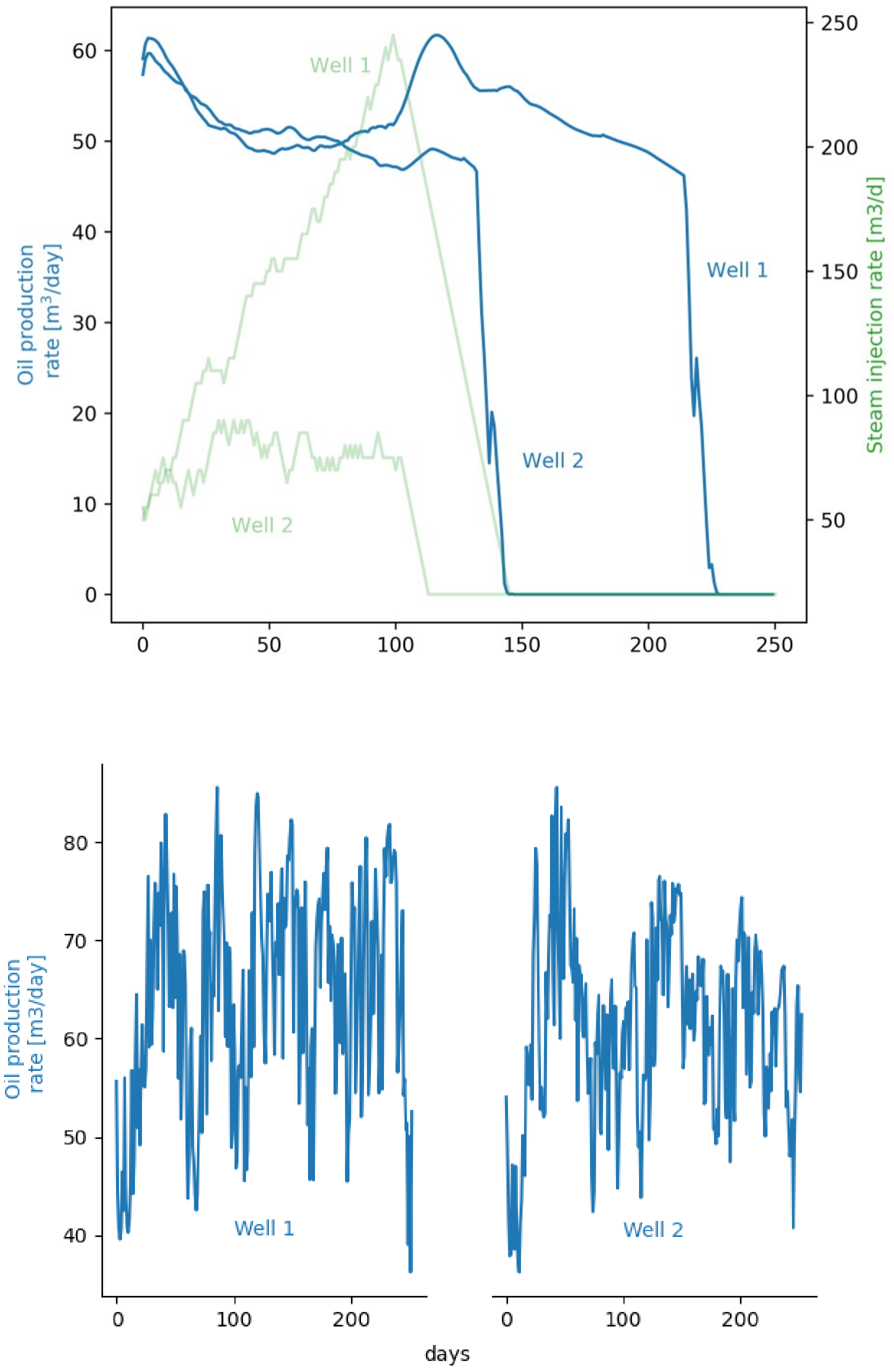


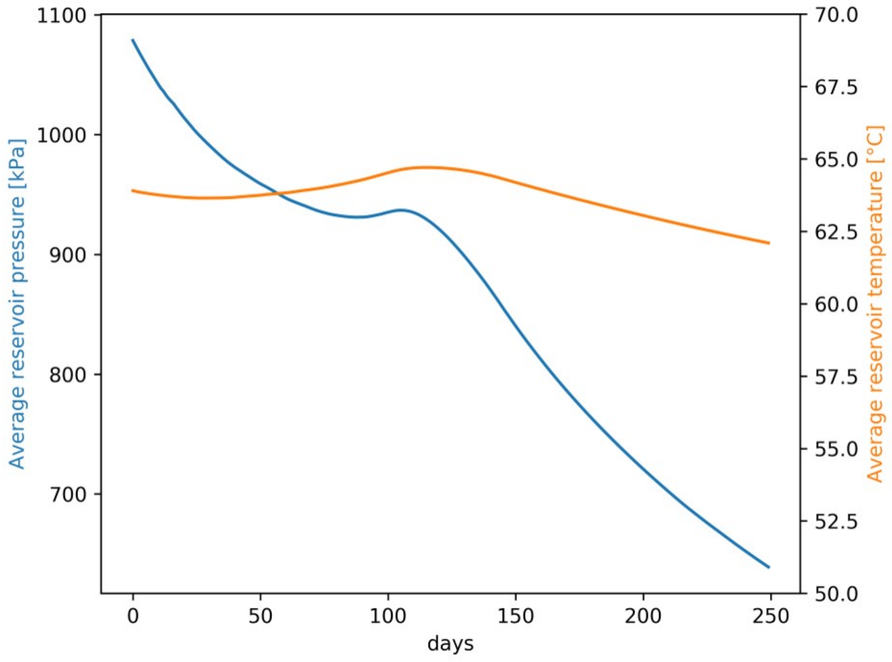
Figure 4.7: Oil production and steam injection rates of optimal case

tain the reservoir pressure. Note how, the increasing steam injection rates observed in Region 1, the agent manages to make the reservoir pressure reach a saddle point at around 925 kPa at exactly t_{drop}. Afterwards, reservoir pressure drops continuously until the end of the production horizon, but based on Figure 4.8-top, oil production drops much later. Regarding average reservoir temperature, we can see that it suffers no significant change throughout the episode; it remains in values of between 61 to 65 °C. We see a different picture when analyzing the CMOST-obtained optimal policy in (Figure 4.8-bottom). Notice how on one hand, the average reservoir pressure increases significantly for the first 50 days of production reaching values of around 1350 kPa and then decreasing to around 1200 kPa. The average reservoir temperature on the other hand, increases steadily throughout the production horizon from an initial value of 64 °C to 72 °C.

4.5 Summary

In this work, a policy gradient strategy was used to find the optimal steam injection policy for a two-well SAGD process. In particular, this optimal steam injection policy maximizes cumulative net present value at the end of the production horizon. A reservoir simulation model inspired by a northern Alberta reservoir represents the environment and deep neural networks were used to parametrized the policy. Results suggest that:

- Optimal steam injection policy for both wells can be divided in two regions: 1) an increase or slight increase of steam injection rates, and 2) a sharp decrease until reaching the minimum value
- The objective of Region 1 seems to be to keep average reservoir pressure from reducing drastically, in fact, the steam injection rates, manage to make the



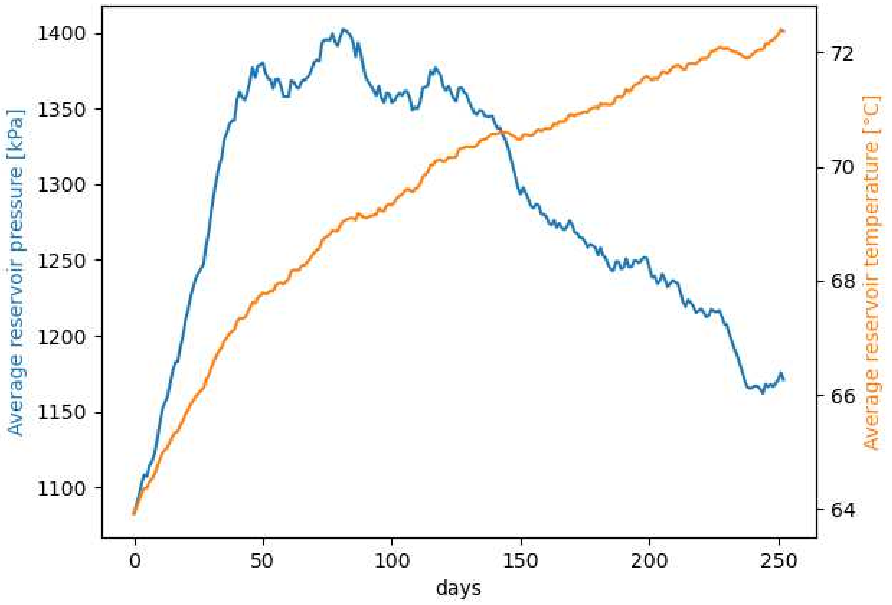


Figure 4.8: Average reservoir pressure and temperature under RL-obtained (top) and CMOST-obtained (bottom) optimal steam injection policy

pressure reach a saddle point, afterwards, reservoir pressure is reduced continuously until the end of production horizon (Region 2)

- In Region 2, the goal appears to be collect high values of reward due to the reduction in steam injection rates while oil production rates stay high

This work represents another step in the direction of a general SAGD multi-well optimization framework that optimal includes steam-allocation, and to contribute to the discussion of identifying key characteristics of optimal policies for SAGD. In this context RL poses as an effective alternative considering that:

- The agent does not require the mathematical model of the reservoir dynamics and thus, the reservoir simulation model is used as a black-box element used to train the agent

- Maximization is performed over the cumulative reward function or net present value. We are only interested in maximizing the net present value at the end of the production horizon (*e.g.* 5 years)

- Using deep neural networks, any policy can be potentially effectively parametrized

Chapter 5

An evaluation of the recursive prediction approach for dynamic surrogate modeling and optimization[1]

5.1 Introduction

In this chapter, we evaluate the performance of the conventional recursive prediction approach to predict key time-varying outputs in a SAGD oil recovery process and its possible use for steam injection optimization. In particular, we study what are the most important features necessary to predict key variables (*e.g.* oil production rates, reward), the robustness of the model(s) with respect to perturbations of these features, the effectivity of the approach or how well can it predict time-varying features, the behavior of the residuals over time, and the impact of the design of experiments over the predictive performance of the approach.

Key findings of this work include:

- The recursive approach is able to properly forecast reward (net present value)

[1][40] A version of this chapter has been presented in the 2022 SPE Western Regional Meeting in Bakersfield, California, USA

very well when considering daily (Case 1) dynamics as opposed to when considering weekly time steps (Case 2)

- The reward model for Case 1, depends mostly on non-predicted or given values (*e.g.* steam injection rates), while for Case 2, depends significantly on predicted values (*e.g.* oil production rates for well 4 and 5)

5.2 Recursive prediction methodology

The objective of surrogate modeling is to identify a function $\hat{\mathbf{g}}$ that can properly describe the environment's dynamic (Equation 2.1) but with a significant reduction in computational cost, *i.e.*

$$x_{t+1} = \mathbf{g}(x_t, u_t) \approx \hat{\mathbf{g}}(x_t, u_t) \tag{5.1}$$

where x_t represents the state of the system at time t, *e.g.* oil/water production rate, average pressure/temperature, oil saturation distribution, etc.

As shown in Figure 5.1, the recursive prediction approach consists of identifying $\hat{\mathbf{g}}$ and then using the state predictions of t, as input for the prediction at $t + 1$. Additionally, at each time step, a new input u_t will be given by the steam injection policy of interest, *e.g.* from the design of experiments, optimizer, or arbitrary policy.

In this work, M surrogate models are identified as represented by Equation 5.2. Each of these M models will capture the dynamics of a part of state x as a function of the complete state x_t and the inputs u_t, *i.e.*

$$\begin{aligned} x_{t+1,1} &= \hat{\mathbf{g}}_1(x_t, u_t) \\ x_{t+1,2} &= \hat{\mathbf{g}}_2(x_t, u_t) \\ &\vdots \\ x_{t+1,M} &= \hat{\mathbf{g}}_M(x_t, u_t) \end{aligned} \tag{5.2}$$

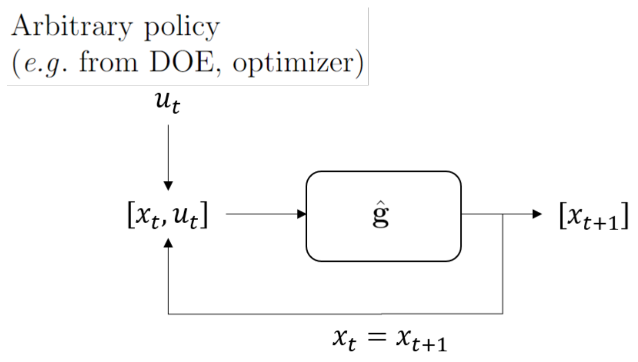


Figure 5.1: Recursive prediction approach

Thus, x_{t+1} will be formed by combining the output of all M models, *i.e.*

$$x_{t+1} = \begin{bmatrix} x_{t+1,1}, & x_{t+1,1}, & \ldots, & x_{t+1,M} \end{bmatrix} \tag{5.3}$$

In this case, the recursive approach starts with a known initial state x_1 and a time series of steam injection rates $\mathbf{u} = [u_1, u_2, \ldots, u_T]$. Now x_1 and u_1 are used as input for the M surrogate models and x_2 is constructed as shown in Equation 5.3. Then, x_2 and u_2 are used recursively to predict x_3, and the process continues. Equation 5.1 attempts to satisfy the Markov property, i.e., the capacity to predict the future using information from the previous time step. This is also called a memory-less property and implies that the definition of state should be complete enough that the history information of the process is unnecessary.

5.3 Surrogate Modeling and Optimization

This section describes the main four (4) steps followed in this work to identify the surrogate models as expressed in Equation 5.1 and the prediction specifics.

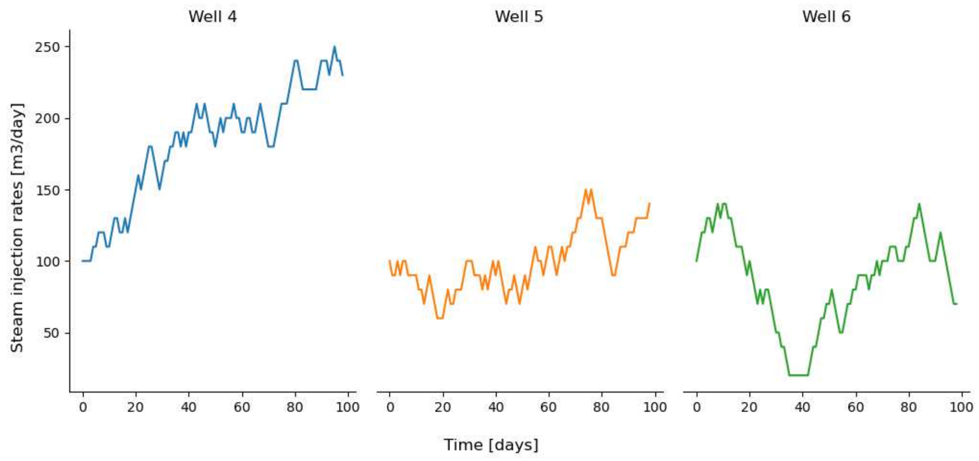


Figure 5.2: Sample of steam injection rates time series for each considered well

5.3.1 Design of Experiments

This step consists of generating the data used for surrogate model identification and validation. In particular, involves selecting a pre-established number of samples, each sample represents a steam injection rates time series for each considered well (Figure 5.2). On one hand, the samples should provide a thorough representation of the complete input space, in other words, it should exhibit low discrepancy. On the other, the evaluation of each sample consists of executing a computationally expensive reservoir simulation model; therefore, the number of samples should be as low as possible.

Well-known design of experiments strategies include Latin Hypercube Sampling [62], Hammersley Sequence Sampling [109], and Sobol Sequence Sampling [50]. These strategies have been successfully applied for dynamic surrogate modeling for nonlinear chemical processes [91] and waterflooding [72]. However, for each sample, the input value at every time step is chosen independently from each other, as a consequence the input time series could exhibit very high changes which could create operational problems for SAGD processes.

To overcome this issue, we propose the use of the RL concept of a discrete action. At every time step, the system must choose an action, represented by an integer number that can be mapped to a specific steam injection rate for each considered well, e.g., Well 4, 5, and 6. At every time step, the steam injection rate at time t+1 for each well has three (3) possible actions: increase/decrease steam injection rate at time t by adding/subtracting a constant value (Δ) or no change. Additionally, because these three possible actions are available for each of the three (3) well pair, the action space consists of all the combinations of actions/well pairs, *i.e.* $3^3 = 27$. For example, one such combination could be: increase (Well 4), no change (Well 5) and decrease (Well 6). So, by setting an initial steam injection rate we can calculate the following steam injection rates by knowing the action number or combination. In this work, the action time series is chosen randomly, i.e., at each time step, a random integer number between 1 and 27 is chosen.

This action strategy has the following benefits:

- Reduces the action space to a single time series of a finite number of actions at each time step
- Prevents drastic changes in the steam injection rates that could bring operational problems
- Does not assume the hydraulic independence of each well pair

5.3.2 Evaluation of each sample using a high fidelity model

Each of the samples generated in the DOE will represent a different numerical reservoir simulation model [19]. This model is represented by, a set of partial differential equations generated from conservation principles (*e.g.* material balance equations- one for each hydrocarbon component, energy balance equation), thermodynamic phase equilibrium conditions, and constraint equations (sum of saturations, oil-phase mole fraction, and gas-phase mole fractions, must be 1).

Two (2) types of border conditions are used, i) Dirichlet condition, by specifying bottom hole pressure specified in producing wells, and ii) Neumann condition, to specify no-flux (mass and heat) boundaries (at the limits of the domain or reservoir) and steam injection rates given by the DOE at a given temperature in the injecting wells. With respect to initial conditions, initial water saturation is determined by irreducible water saturation given by the relative permeability curves. Additionally, water-oil and gas-oil contacts are considered well below and above the oil reservoir, respectively. Initial pressure and temperature are estimated from the value of pressure and temperature at a reference depth (datum); pressure is then calculated using the condition of hydrostatic equilibrium and temperature is considered constant.

5.3.3 Surrogate model identification and validation

For each of the evaluated samples or stem injection policies, at every time step, the necessary information to form x is found. For example, x could be represented by oil production rate, reservoir average pressure, oil saturation distribution. Thus, $\hat{\mathbf{g}}$ is identified to map $x_t, u_t \rightarrow x_{t+1}$ using a conventional supervised learning framework (*e.g.* recurrent neural networks), and the prediction approach described in Section 5.2 is validated.

Two stages of validation are used, the first one corresponds to the quality of the surrogate models with respect to the one-step dynamics, as shown in Equation 5.1. The second is the performance of the surrogate models to predict the time series. This consists of comparing the predicted and true time series output for each part of x_t of each validation sample. For optimization purposes, this output is represented in Equation 2.1 as J.

5.3.4 Optimization

Once one or a set of surrogate model(s) are identified, the execution time of the optimization methods described above (e.g., RL, MPC, genetic algorithms) is no longer an obstacle. This optimization problem can be formulated as,

For a given SAGD configuration with n injector and producer wells (pairs) and a given production horizon with T time steps (*e.g.* daily), the objective is to find the steam injection policy that maximizes a particular performance measure (*e.g.* net present value) while accounting for the complex reservoir dynamics. Formally, the optimization problem can be written as:

$$\begin{aligned} \max_{\mathbf{u} \in \mathbb{R}^{n \times T}} \quad & J = \sum_{t=1}^{T} J_t(x_t, u_t) \\ \text{s.t.} \quad & \mathbf{u}_{min} \geq \mathbf{u} \geq \mathbf{u}_{max} \\ & x_{t+1} = \hat{\mathbf{g}}(x_t, u_t) \end{aligned} \tag{5.4}$$

where J is a scalar performance measure, $u_t \in \mathbb{R}^{n \times 1}$ is the steam injection rates for all n wells at time t, x_t is the state of the SAGD dynamic system at time t, $\mathbf{u}_{min}$ and $\mathbf{u}_{max}$ are the upper and lower bounds of the steam injection rate, and $\hat{\mathbf{g}}$ is the surrogate model of the SAGD dynamic system. Typical J functions include steam-oil ratio, oil recovery factor, and net present value.

5.4 Case Study

This section describes the numerical reservoir simulation model that describes the SAGD dynamics based on first principles, *surrogate modeling* specifics, i.e., the definition of state - x_t, type and architecture of surrogate, training specifics, and a brief description of the *optimization* algorithm.

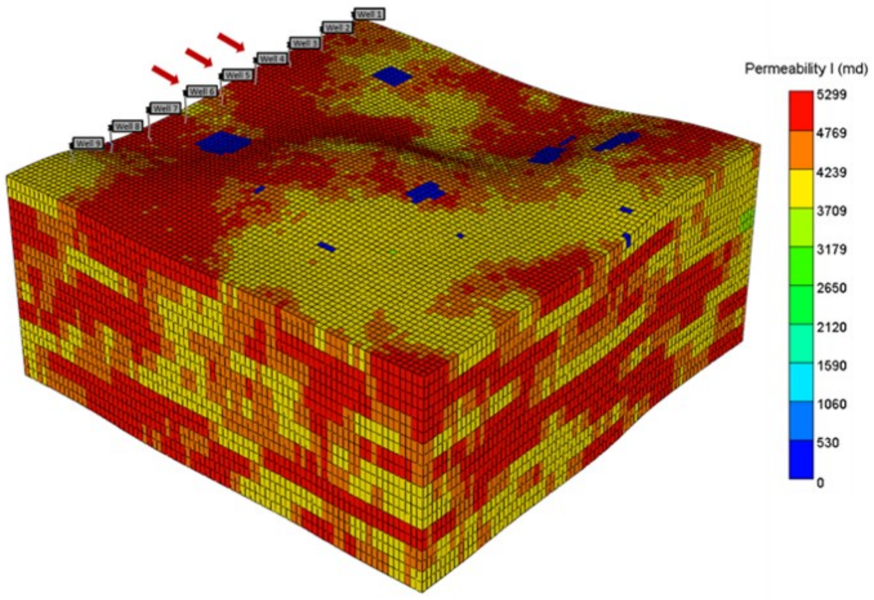


Figure 5.3: 3D grid and porosity field with indication by a red arrow of the considered wells (case study)

5.4.1 Reservoir simulation model

A synthetic reservoir simulation model was created using CMG-STARS 2018.10 and based on publicly available data of heavy oil SAGD projects. In particular, average porosity and permeability were taken from data from the Christina Lake Thermal Project [12] in Canada, and thermal properties, fluid components, and rock-fluid data were taken from heavy oil models used for CMG software [19] training purposes. The model consists of nine well-pairs but only three (3) are considered for this work, i.e., Wells 4, 5, and 6 indicated by a red arrow in Figure 5.3; the other wells are shut during the surrogate modeling time considered. Additionally, shale bodies were distributed using object modeling geostatistics to reflect typical rock heterogeneity found in heavy oil reservoirs.

Average reservoir, thermal properties, grid data, and the fluid component definition can be found in Tables 5.1, 5.2 and 5.3. Specifically, the grid consists of 80

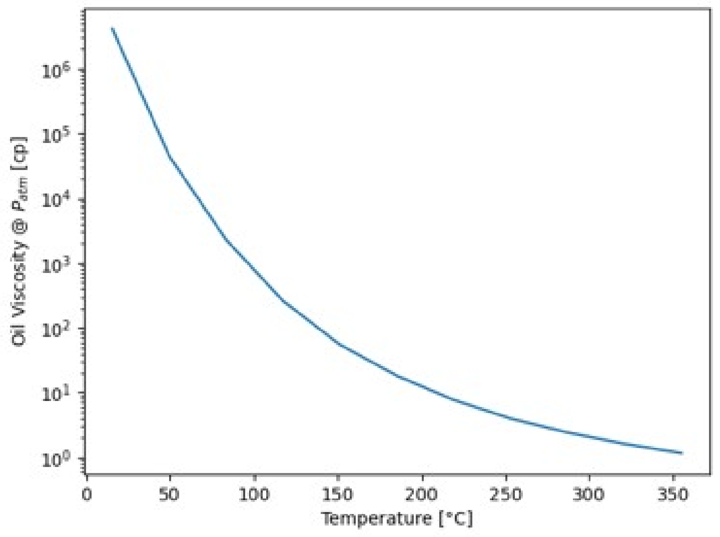


Figure 5.4: Oil viscosity with respect to temperature (case study)

x 80 x 20 grid blocks in the i, j, k with grid size 800 m, 800 m, 50 m, respectively. The model exhibits an initial oil and water saturation of 0.74 and 0.19, respectively, and an initial pressure of 2011 kPa and a temperature of 48.8°C. Two (2) sets of relative permeability curves are used, one for the shale bodies and one for the sand or the rest of the reservoir. These curves are shown in Figure 5.5 and changes in oil viscosity with respect to temperature are shown in Figure 5.4.

5.4.2 State definition and training details

The input or state x_t consists of 19 elements shown in Table 5.4, this definition is arbitrary but has to provide the model with Markov property. As referred in Equation 5.1, x_t and u_t will allow the forecast of x_{t+1}. To this end, five different deep long-shot-term-memory neural networks (LSTM-NN) are used as is illustrated in Figure 5.6. There is one individual LSTM-NN for the prediction of oil production rates, water production rates, average pressure around wells, the average temperature

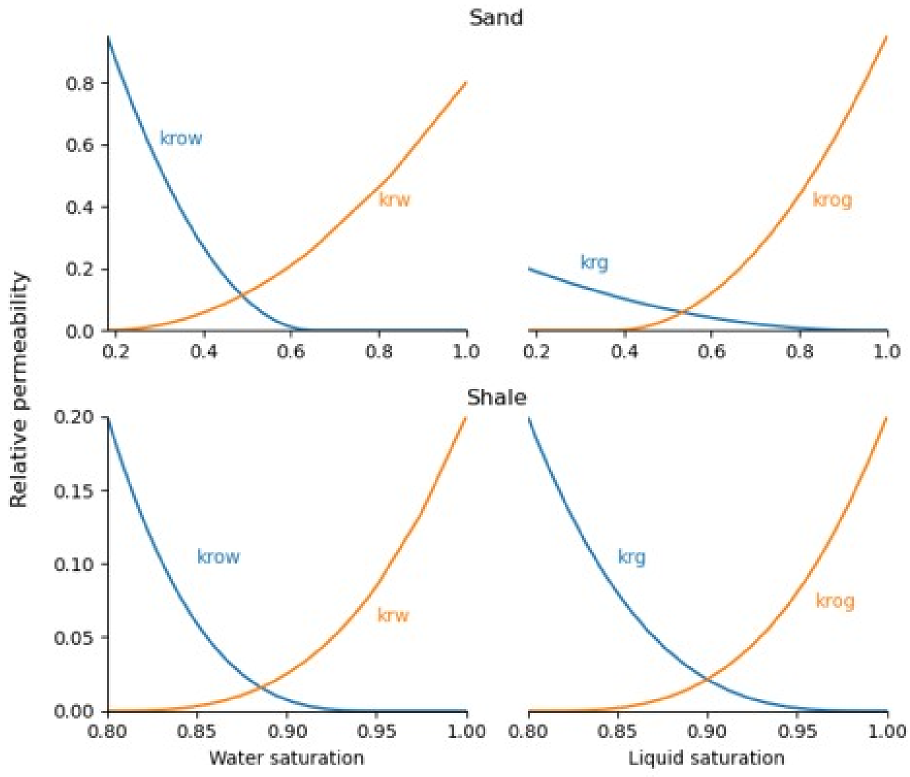


Figure 5.5: Relative permeability curves for sand (upper) and shale (lower) – (case study)

Table 5.1: Reservoir simulation model parameters (case study)

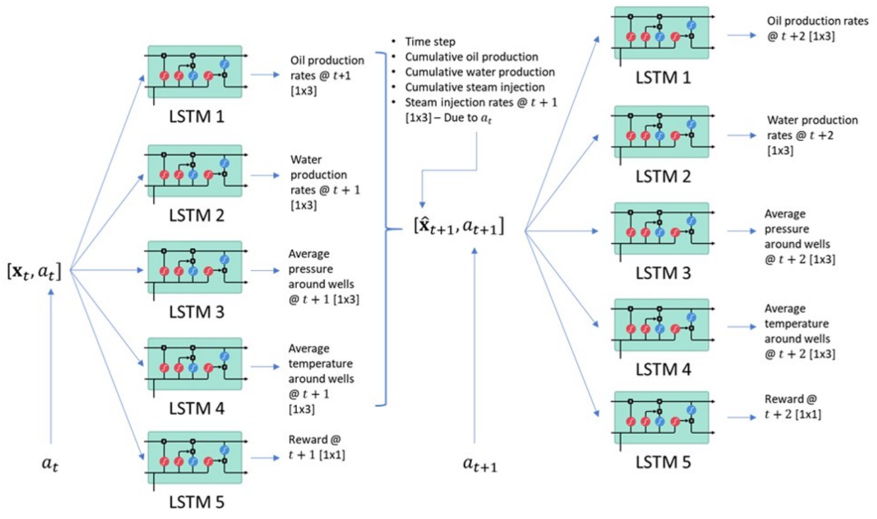
Reservoir data	Reservoir compressibility [kPa^{-1}]	1.00×10^{-6}
	Rock heat capacity [$J/m^3 °C$]	2.3×10^6
	Rock thermal conductivity [J/m day°C]	2.7×10^5
	Reservoir initial temperature [°C]	48.4
	Reservoir initial pressure [kPa]	2011
	Initial water saturation	0.19
	Initial oil saturation	0.74
	Average porosity	0.29
	Average permeability [mD]	4275
	Well spacing [m]	80
Production parameters	Production pressure [kPa]	1000
Production horizon/episode length [d]		250

Table 5.2: Grid data (case study)

	x	y	z
Number of gridblocks	80	80	20
Gridblocks size [m]	10	10	10
Reservoir size	800	800	50

	H_2O (aqueous)	Heavy (oleic)	CH4 (oleic)
Molecular weight [kg/g − mole]	0.018	0.6	0.016
Critical pressure [kPa]	21997	939.72	4600.15
Critical temperature [°C]	374	793.73	−82.55
Mass density [kg/m^3]	998	1012.8	351.68
Liquid compressibility [kPa^{-1}]	0	2.307×10^{-7}	3.532×10^{-6}
Thermal expansion coefficient (1st coeff) [$°C^{-1}$]	0	2.217×10^{-5}	0.00149

Table 5.3: Component definition (case study)



$[\mathbf{x}_t, a_t]$
a_t
LSTM 1
LSTM 2
LSTM 3
LSTM 4
LSTM 5
Oil production rates @ t+1 [1x3]
Water production rates @ $t+1$ [1x3]
Average pressure around wells @ $t+1$ [1x3]
Average temperature around wells @ $t+1$ [1x3]
Reward @ $t+1$ [1x1]
• Time step
• Cumulative oil production
• Cumulative water production
• Cumulative steam injection
• Steam injection rates @ $t+1$ [1x3] – Due to a_t
$[\hat{\mathbf{x}}_{t+1}, a_{t+1}]$
a_{t+1}
Oil production rates @ t +2 [1x3]
Water production rates @ t +2 [1x3]
Average pressure around wells @ $t+2$ [1x3]
Average temperature around wells @ $t+2$ [1x3]
Reward @ $t+2$ [1x1]

Figure 5.6: Illustration of the recursive prediction process using five (5) LSTM neural networks (case study)

around wells, and reward. After each time step, the state is constructed based on the prediction of each of these LSTM-NN on one hand, and on-time step counter, the on-going calculations of the cumulative oil and water production and steam injection and on the steam injection rates calculated by the mapping procedure detailed in the previous section. This new state is fed to the same LSTM-NN and the process moves to the following time step.

In this case reward (LSTM 5) is a variable that represents a scalar performance measure exhibited by the reservoir after taking an action. Although it is not part of the state it is used for optimization purposes as the objective function - J, i.e., maximizing cumulative reward. In this work, the reward is defined as net present value (Equation 5.5) using the parameters shown in Table 5.5.

$$J_t = \frac{P_o \sum_{j=1}^{n} q_o, j - C_s \sum_{j=1}^{n} q_s, j - C_w \sum_{j=1}^{n} q_w, j}{1 + i^{\frac{t-t_{ref}}{365}}} \tag{5.5}$$

Two (2) different sizes of time steps are used to evaluate the accuracy of the

Table 5.4: State definition (case study)

Time step	Water production rate for W4
Cumulative oil production	Water production rate for W5
Cumulative water production	Water production rate for W6
Cumulative steam injection	Average pressure around W4
Steam injection rate for W4	Average pressure around W5
Steam injection rate for W5	Average pressure around W6
Steam injection rate for W6	Average temperature around W4
Oil production rate for W4	Average temperature around W5
Oil production rate for W5	Average temperature around W6
Oil production rate for W6	

Table 5.5: Economical parameters (case study)

Parameter	Value
P_o [USD/STB]	80
C_{steam} [USD/STB]	12
C_{water} [USD/STB]	6
i [fraction]	0.1
t_{ref} [day]	0

recursive prediction process: daily (Case 1) and weekly (Case 2) with a production horizon of 250 days and 100 weeks, respectively. The objective is to study the impact of the different time-scale on the capacity of the LSTM-NN to fully capture the dynamic of the system. This is particularly important considering SAGD is a time-lagged, slow, and non-linear phenomenon. Regarding the number of samples, 100 and 200 samples were used, for Case 1 and Case 2, respectively and in both cases, an 80-20 training-validation split was used.

5.4.3 Surrogate modeling - LSTM

The formulation expressed in Equation 5.1 describes a time series prediction problem that has been successfully modeled using Long-Short-Term Memory (LSTM) neural networks. LSTM networks represent a special type of Recurrent Neural Networks (RNN), ones that are specially designed to capture the dynamics of sequence data. Furthermore, LSTM networks can mitigate the so-called vanishing gradient problem, typical in conventional RNNs that prevent them from being able to capture long-term dynamics. The mechanics of LSTM networks relies on the concept of a cell state c_t and three gates: forget, input, and output (Figure 5.7). The idea is that the gates will remove or add information from the cell state. The first gate or the forget gate controls the information from the cell state that will be removed or forgotten. This gate receives $h_{(t-1)}$ or the previous hidden state and the input x_t and applies the sigmoid function, *i.e.*

$$f_t = \sigma(W_f \cdot [h_{t-1}, [x_t, u_t]] + b_f) \tag{5.6}$$

and the multiplication operator is applied to the cell state c_{t-1}. Intuitively because of the sigmoid function f_t will be a vector of 0s and 1s, hence when a value is 0 means the information is forgotten.

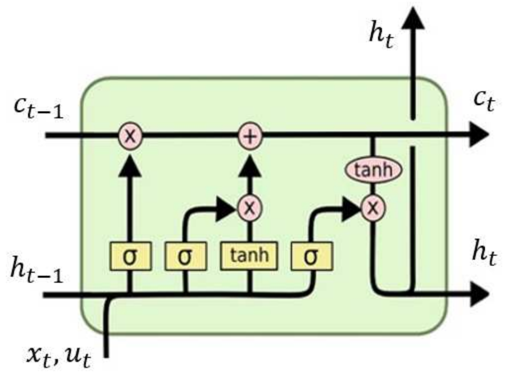


Figure 5.7: LSTM architecture. Figure credits: [74]

The second step is the input gate, here we will decide what new information to add to the cell state. First, the input vector h_{t-1}, x_t is again passed through a sigmoid function as described by Equation 5.6. Second, the input vector is passed through a tanh function; the result will be a value between -1 and 1. These two values will be multiplied and added to the cell state, *i.e.*

$$\begin{aligned} i_t &= \sigma(W_i \cdot [h_{t-1}, [x_t, u_t]] + b_i) \\ \tilde{c}_t &= \tanh(W_c \cdot [h_{t-1}, [x_t, u_t]] + b_c) \end{aligned} \tag{5.7}$$

Intuitively after applying the sigmoid function, the information that will be added to the cell state is selected. Because the result will be 0 and 1, the values corresponding to 1 are values that will be updated. The second part will represent the scale as to how much to update these values. At this point, the cell state is updated as,

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t \tag{5.8}$$

Finally, the output gate decides the output; this will be a filtered version of the cell state. The first part consists of again applying a sigmoid to the input vector, here we will decide what information of the cell state will correspond to the output. The second part entails applying a tanh function this time to the cell state and then multiplying both of these terms, *i.e.*

$$\begin{aligned} o_t &= \sigma(W_o \cdot [h_{t-1}, [x_t, u_t]] + b_o) \\ h_t &= o_t \cdot \tanh c_t \end{aligned} \tag{5.9}$$

For both Case 1 and Case 2, deep neural LSTM-NN with a variable number of intertwined fully connected, and dropout layers are used. In Case 1, the number of layers (fully connected/dropout) is 3 for all variables except reward, in which 8 layers were used. For Case 2, the number of layers (fully connected/dropout) is 7 for all variables except oil rates, in which 2 layers were used. In all cases, the number of neurons used in each layer was 50, the dropout rate was set at 0.2, the loss function used was mean square error, and the Adam optimizer was used during the training process. Implementation was done in Python 3.8 using TensorFlow [60] libraries.

5.4.4 Surrogate model validation

As mentioned in Section 5.3.3, two (2) stages of validation are carried out, the first one corresponds to the quality of the surrogate models with respect to the one-step dynamics, and the second is the performance of the recursive approach to predict the time series of interest.

For the first stage, $R2$ is used, as defined by:

$$R2_k(y, \hat{y}) = 1 - \frac{\sum_{t=1}^{T} (y_t - \hat{y}_t)^2}{\sum_{t=1}^{T} (y_t - \bar{y}_t)^2} \tag{5.10}$$

where y_t represents the true values of the variable of interest (see Table 5.4) at time t as obtained via the numerical reservoir simulation model (Section 5.3.2), $\hat{y}_t$ is the prediction made by the LSTM neural networks, $\bar{y}_t$ the arithmetic average of the true values, and T is the prediction horizon.

Additionally, feature importance and robustness test are performed on a selection of the identified one-step dynamics models. The feature importance test corresponds to the methodology implemented in [76], in which initially a reference $R2_{base}$ is estimated on the model using the test data. Afterward, each feature (Table 5.4) is randomly shuffled to generate a corrupted version of the test data and a new $R2_{corrupted}$ value is calculated. This procedure is repeated K times and the importance of each feature j is then estimated as:

$$i_j = R2_{base} - \frac{1}{K}\sum_{k=1}^{K} R2_{corrupted,k,j} \tag{5.11}$$

The robustness test consists of corrupting each feature by adding a noise drawn from a normal distribution of mean zero and varying sigmas. Sigmas are varied from 0 (uncorrupted case) to 0.8 in increments of 0.1, and for each sigma, a perturbed value of $R2$ is calculated.

For the second stage, the true and predicted time series are also validated using Equation 5.10 for five (5) different DOEs to test the effect of the randomness of the design of experiments on the results. Additionally, the residuals with respect to time step prediction are also evaluated. This residual is defined as,

$$Residual = |y_i - \hat{y}_i| \tag{5.12}$$

This error is calculated at each time step of the time series; thus, we will have a time series of errors for each of the test data samples. Thus, the mean and variance of the residuals are evaluated as a function of time.

5.4.5 Optimization - Genetic algorithms

Once one or a set of fast surrogate model(s) is identified, the execution time of the optimization methods described above (e.g., RL, MPC, genetic algorithms) is no longer an obstacle. In this work, we use genetic algorithms.

Genetic algorithms represent a family of optimization strategies inspired by the theory of natural evolution in which the reproduction of the fittest individuals will produce even fitter offspring. In this sense, a steam injection time series (Figure 5.2) is considered an individual; in particular as explained in the Design of Experiments section, a sample is represented by a string of actions (integer numbers between 1 and 27). Each of the actions in the string is referred to as a *Gene*, and the string of Genes are known as a *Chromosome*; typically, the genes are encoded in binary values. In general, the algorithm can be separated into three (3) phases:

1. **Initialization**. Given an initial population consisting of a predetermined number of individuals, a fitness function is used to determine how fit each individual is. In this case, the fitness function is represented by Equation 1 and represents an evaluation of the identified surrogate model(s)

2. **Creating the next generation**.The individuals for the next generation are created in three ways:

    - *Elitism*. Individuals with the best fitness score are directly selected to survive to the next generation
    - *Crossover*. These individuals are the product of the reproduction between two parents. They are a result of the combination of the parent's genes
    - *Mutation*. Created by generating random changes to selected individuals

3. **Termination**. As this process is repeated, new generations with potentially better fit individuals are generated. Typical termination conditions include, a pre-established number of generations (iterations) is reached, the fitness score exhibited by the best individuals in one generation is not significantly different with respect to the previous generation, etc.

A Python-based implementation was used [83] using the following parameters:

Table 5.6: Genetic algorithm parameters (case study)

Parameter	Value
Maximum number of iterations	100
Mutation probability	0.2
Elite ratio	0.03
Crossover probability	0.7
Parents portion	0.2
Crossover type	Uniform

5.5 Results and discussion

In this section, we present the results and discussion of the surrogate validation as described in Section 5.4.4 and optimization results.

5.5.1 First stage validation

Figure 5.8 shows the distribution of R2 values for the one-step prediction models of reward, oil/water production rates, and average pressure/temperature for Case 1 and 2. Note how in both cases, all the prediction models are able to capture the one-step dynamics very well; Figure 5.9 shows a representative example of this fit. In particular, we see that the median of R2 ranges from 0.925 (reward model for Case 2) to 0.99 (average temperature for Case 2).

Figures 5.10 and 5.11 exhibit the results of the feature importance and robustness test carried out on the reward model and the oil prediction rate for well 5 for Case 1 and 2, respectively. For Case 1 (Figure 5.10) we can see that the most important features for the reward model are in order of importance: steam injection well 2, oil production well 2, steam injection well 1, steam injection well 3, cumulative steam production, cumulative water production, and water production well 2. Additionally,

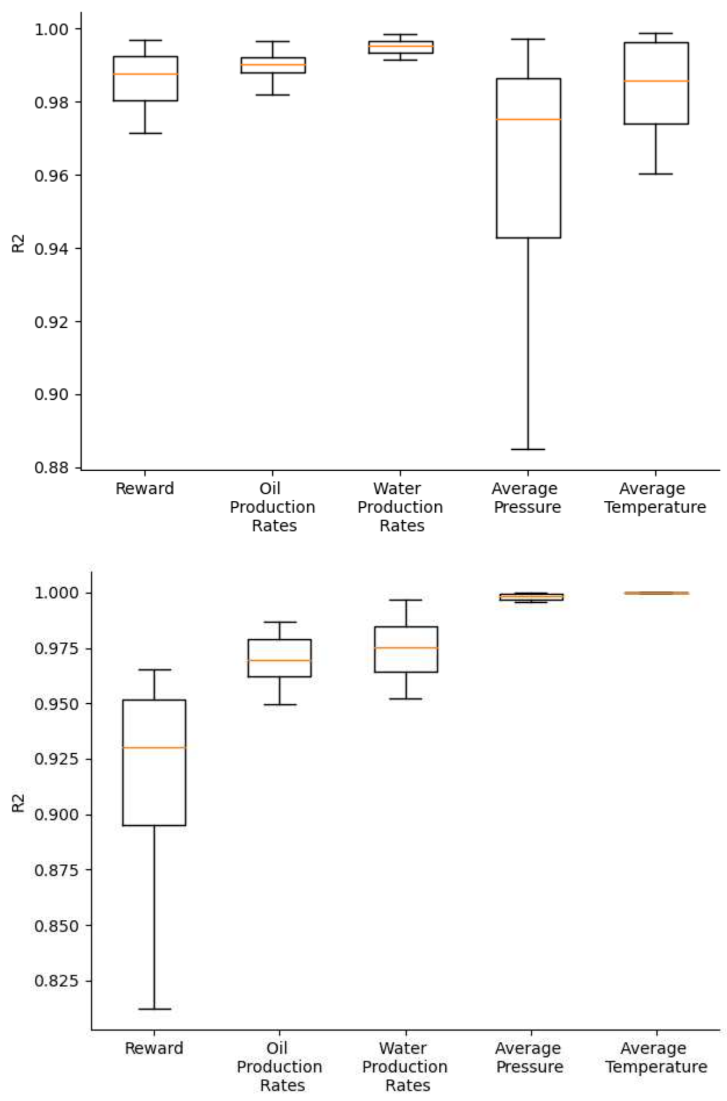
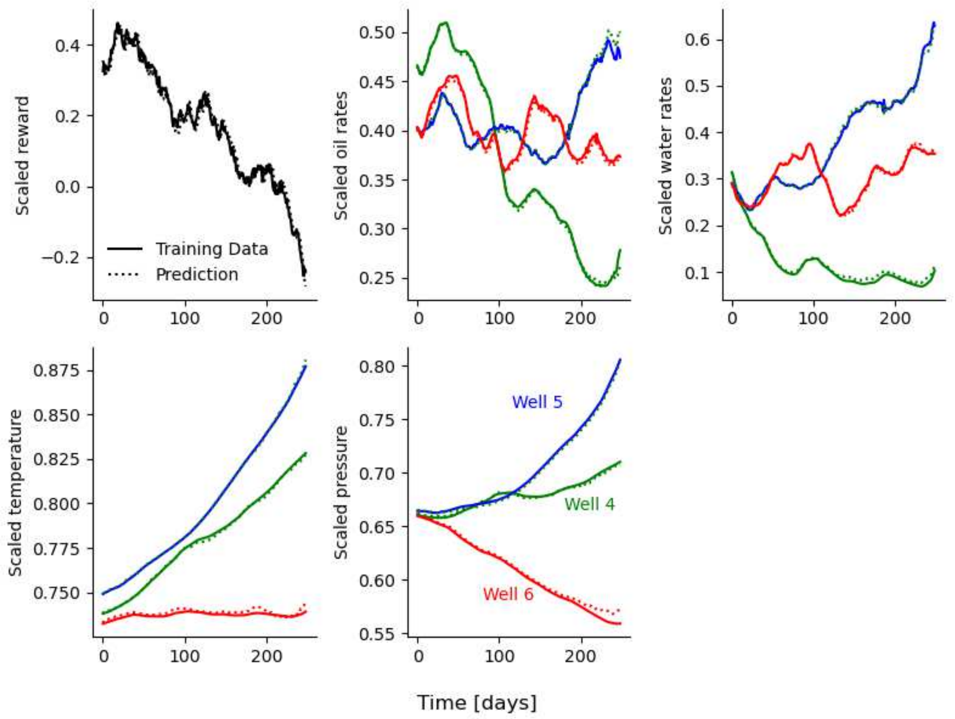


Figure 5.8: R2 values for the one-step prediction of reward, oil, and water production rates, and average pressure and temperature using test data for Case 1 (top) and Case 2 (bottom)



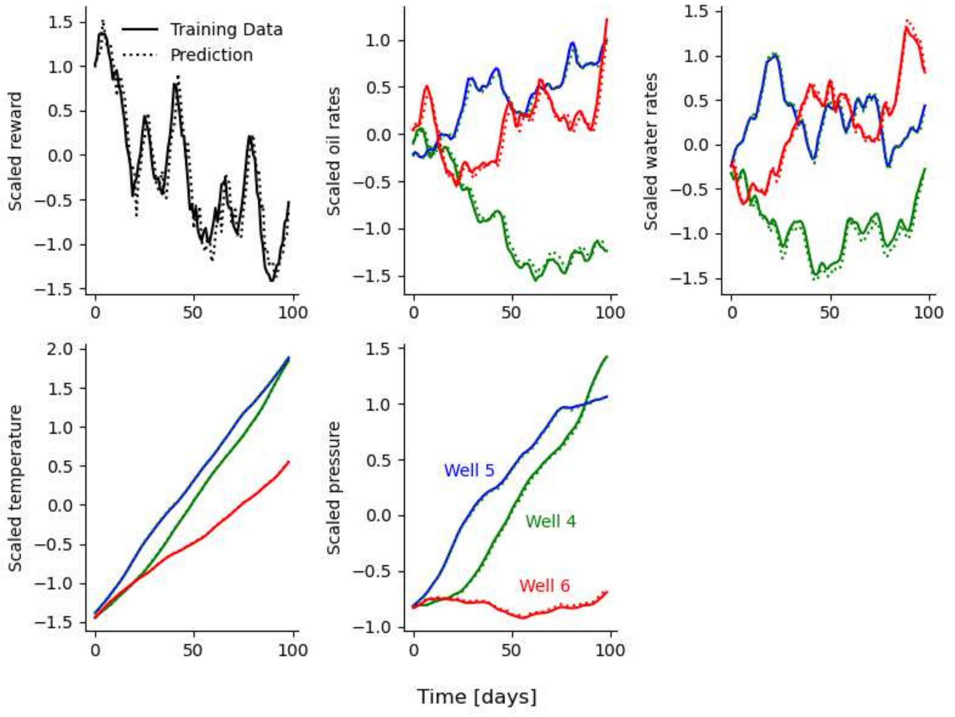


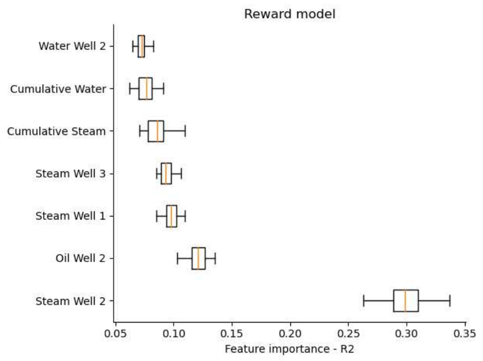
Figure 5.9: Example of fit of one-step models of a representative sample of the test data for Case 1 (top) and Case 2 (bottom).

we can note that the perturbation of these parameters all have a great impact on the reward output. In particular, steam injection rate well 1 and 2 with the greatest impact, while water injection rate well 2 and oil rate well 2 with the least impact.

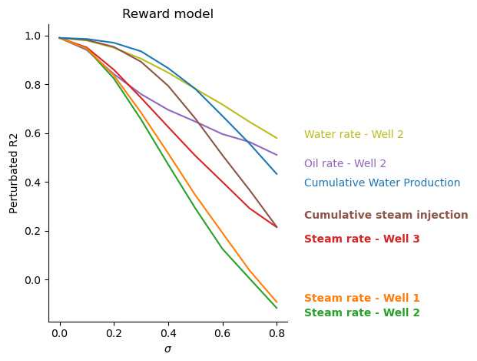
The oil production rate model for well 5 exhibits a very different behavior as can be seen by Figure 5.10. In this case, the previous value of oil production rate for well 5, the cumulative water production, and cumulative oil explain most of the variability of the output. However, variations or perturbations of oil production rate for well 5 have a disproportionately bigger impact on the output than the rest of the features.

These results suggest that the recursive prediction of reward will be much more accurate for the reward model than the oil prediction rate model for well 5. On one hand, the reward model depends much less on predicted features, these are, oil production rate well 2 and to a lesser extent, on the cumulative water production and water production rate for well 2. While the oil prediction model, depends heavily on the previous values of oil production rates, moreover, the model can be greatly influenced by variations of this feature.

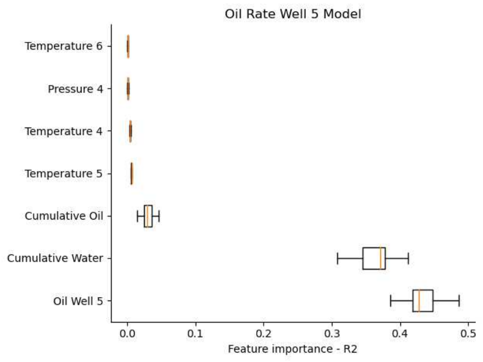
Regarding Case 2 (Figure 5.11) the reward model relies mostly on oil production well 5, steam injection well 5, steam injection well 4, steam injection well 6, oil production well 4, cumulative oil production and cumulative steam injection. Furthermore, variations of most of these features seem to significantly affect the prediction capacities of the model. It is particularly important that unlike for Case 1, the reward model seems to depend heavily on two predicted features: oil production rates for well 4 and 5. This suggests that recursive prediction will be more difficult for the reward model in Case 2 than for Case 1. Regarding the oil production rate model for well 5, we see a similar result to the one seen in Case 1. In particular, the most important feature is previous values of oil production rate for well 5, and variations of this feature have a disproportionately large effect on the output.



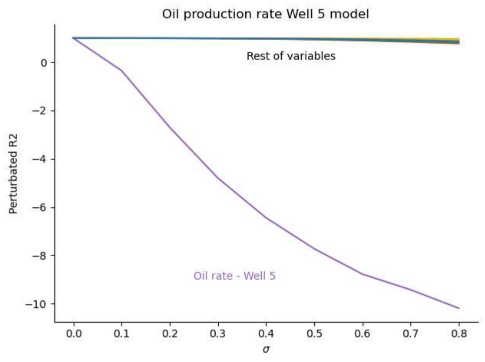
(a) Reward model



(b) Reward model

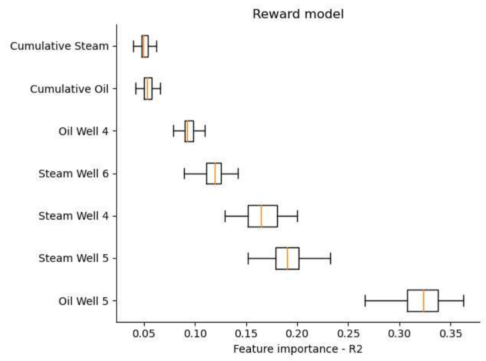


(c) Oil production rate Well 5 model

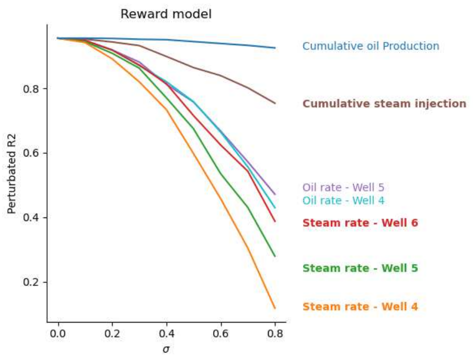


(d) Oil production rate Well 5 model

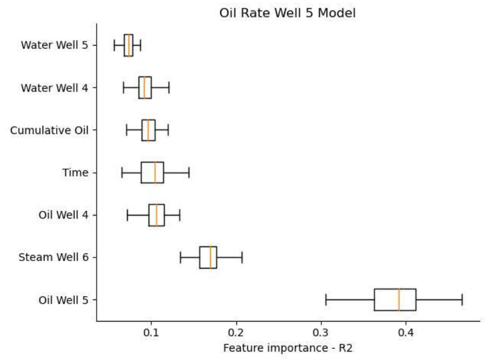
Figure 5.10: Feature importance and robustness test for the reward and oil production rate well 5 models for Case 1



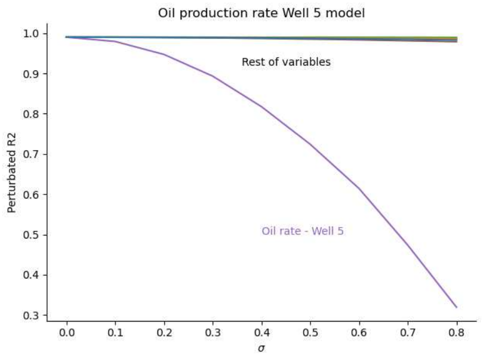
(a) Reward model



(b) Reward model



(c) Oil production rate Well 5 model



(d) Oil production rate Well 5 model

Figure 5.11: Feature importance and robustness test for the reward and oil production rate well 5 models for Case 2

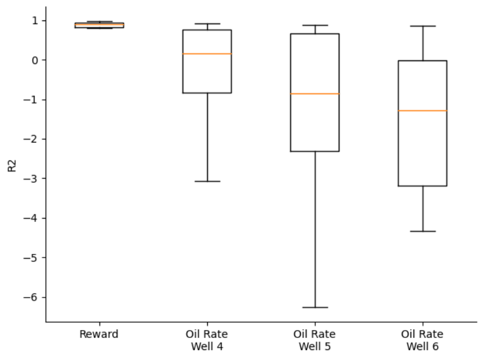
5.5.2 Second stage validation

The performance of the recursive prediction framework for reward and oil production rates for wells 4, 5 and 6 is shown in Figure 5.12. Here, we can see the distribution of R2 values obtained using the test data for Case 1 and 2; a sample of the R2 values for the reward model in shown in Figure 5.13. We can see how reward prediction for Case 1 is significantly superior to oil production rates. In particular, reward prediction shows a median of R2 values of around 0.9, while oil production rates exhibit medians of between 0.3 to -1.2. For Case 2, we see that all the studied variables show very poor median R2 values: -0.88 for reward, -1.33 for oil rate well 4, -1.63 for oil rate well 5, -2.15 for oil rate well 6.

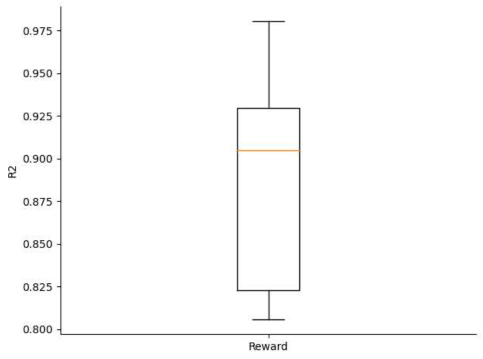
The evaluation of the residuals that result from the recursive prediction framework is shown in Figure 5.14. Note how, the residuals for the reward model in Case 1 are lower than the rest of the variables in Case 1 and all of the considered variables (reward and oil production rates) in Case 2. Moreover, the changes over time of the residuals exhibited by the reward model in Case 1 are much lower than the rest. This can also be confirmed by averaging these residuals, as shown in Figure 5.15. On one hand, this Figure shows that the time average mean of residuals shown by the reward model in Case 1 is around 0.07, lower than the rest of the models in Case 1. On the other hand in Case 2, reward exhibits a time average of residuals of around 0.22, while average temperatures of well 4 and 5 have the lowest residual.

These can be further examined in Figure 5.16, where the residuals are shown over time. We can confirm how the reward model for Case 1 shows very low residuals over time, similar to perhaps to water production rate well 4 and well 5. For Case 2, the lowest residuals are exhibited by average temperature for well 4 and 5 and water production rate for wells 5 and 6.

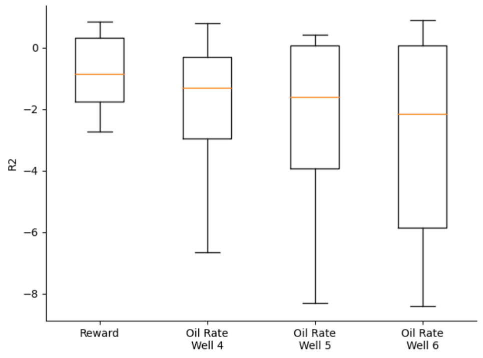
Finally, the evaluation of the impact of the randomness of the DOE over the solution is shown in Figure 5.17. In this Figure, we see the distribution of R2 values



(a) Case Study 1

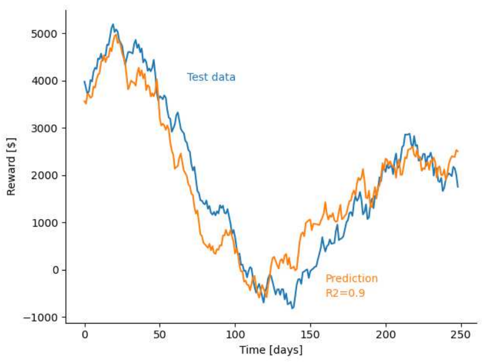


(b) Zoom view of the R2 distribution for the reward model in Case 1

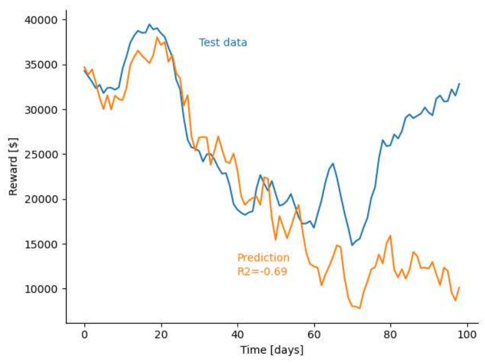


(c) Case Study 2

Figure 5.12: Distribution of R2 values for reward, oil rate - well 4, 5, 6 models using the test data for Case 1 and 2

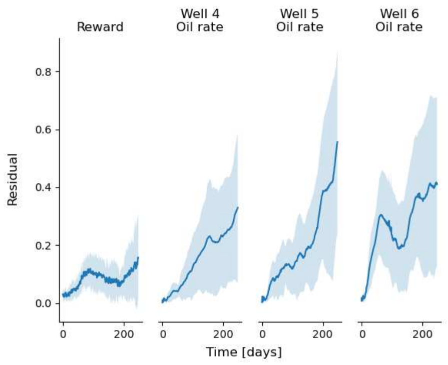


(a) Case Study 1

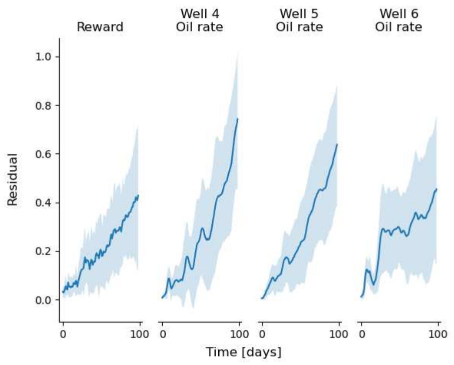


(b) Case Study 2

Figure 5.13: Representative fit of the reward model for Case 1 and 2 with an indication of R2 value

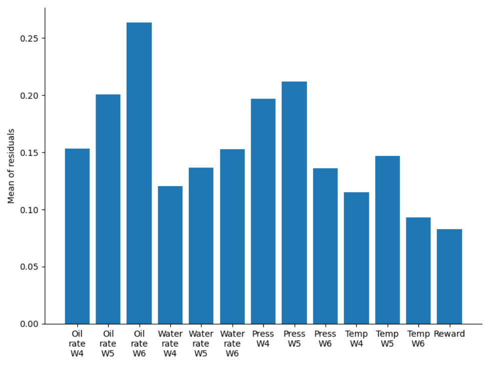


(a) Case Study 1

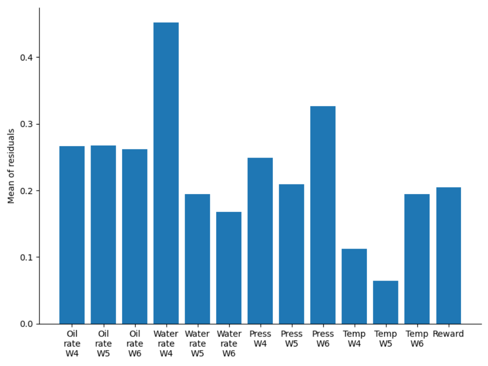


(b) Case Study 2

Figure 5.14: Change of residuals over time for reward, oil rate - well 4, 5, 6 models using the test data for Case 1 and 2

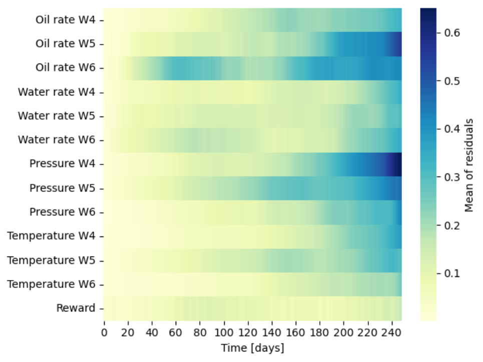


(a) Case Study 1

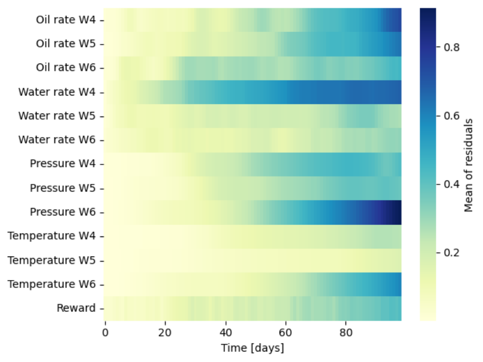


(b) Case Study 2

Figure 5.15: Time average mean of residuals for all models using test data for Case 1 and 2

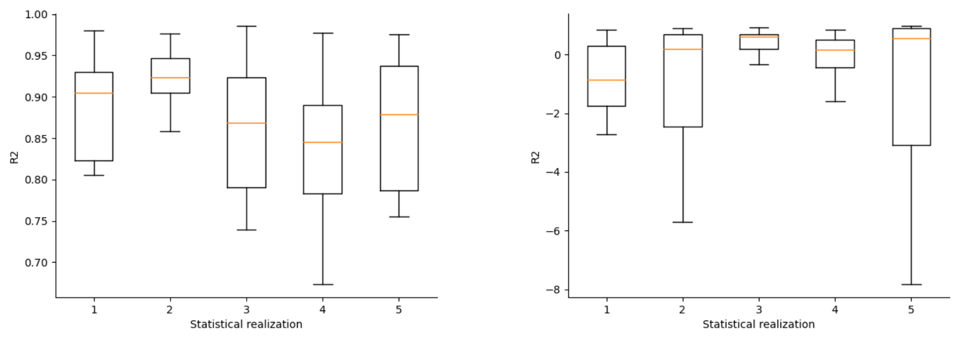


(a) Case Study 1



(b) Case Study 2

Figure 5.16: Mean of residuals over time for all models using test data for Case 1 and 2



(a) Case Study 1 (b) Case Study 2

Figure 5.17: Distribution of R2 values exhibited by the reward model for five different seeds over the test data for Case 1 and 2

exhibited by the reward model for five (5) different statistical realizations of the designs of experiments. On one hand, we can confirm that the great results shown in Case 1 by the reward model can be found consistently in the different seeds with median R2 values ranging from 0.86 to 0.93. On the other hand, for Case 2, we see that the median value of R2 has a high variability, taking -0.88, 0.14, 0.61, 0.14, and 0.56.

5.5.3 Optimization

Based on the performance exhibited by the reward model for Case 1, we can continue to the optimization phase. To this end, Figure 5.18 shows the optimal steam injection rates time series found; notice how in general steam injection rates reduce from the initial state. The rates seem to take low values throughout the production horizon oscillating between 20 and 60 m3/day. Additionally, Figure 5.19 shows the comparison between the true and predicted value of reward in the optimal case. We can see how the prediction is very close to the true value, exhibiting an R2 value of 0.94.

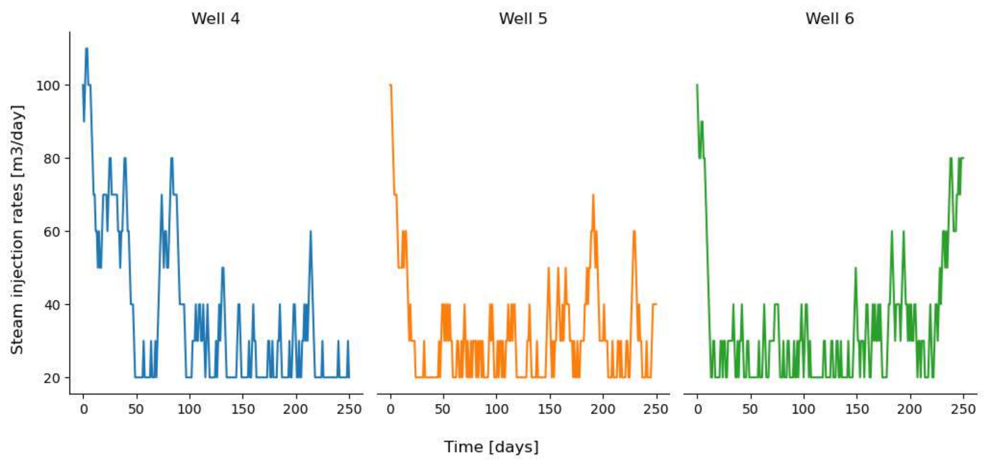


Figure 5.18: Optimal steam injection rates

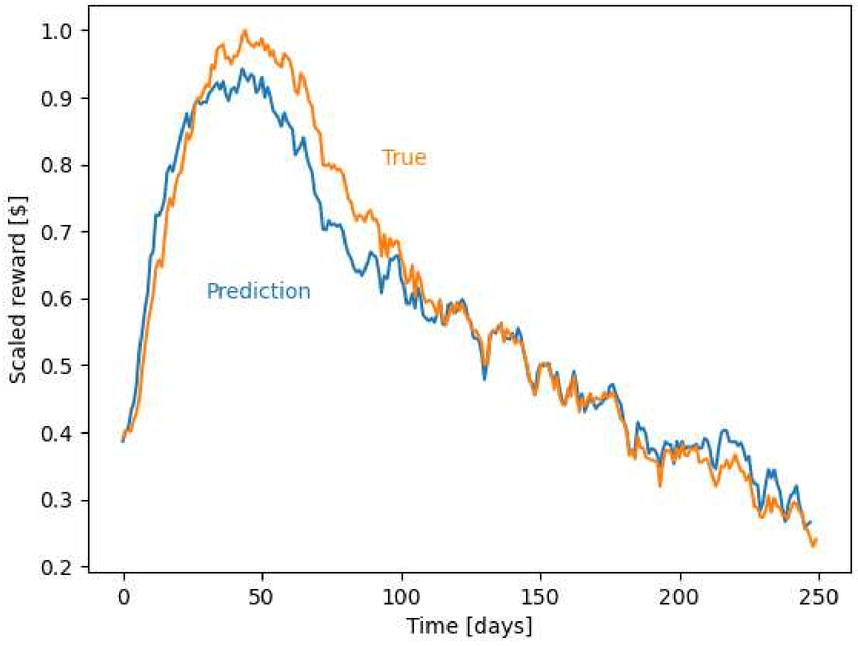


Figure 5.19: Comparison between true and predicted value of reward in the optimal case

5.6 Summary

This work presents a dynamic-surrogate modeling approach to the optimization of steam injection rates for multi-well SAGD processes. The approach relies on the use of Long-short term memory (LSTM) neural networks to capture one-step (daily – Case 1 and weekly – Case 2) dynamics of the phenomena and then use these networks in a recursive scheme to predict multiple steps in the future (time series). The approach was tested using three SAGD well-pairs of a multi-well reservoir simulation model, built using publicly available data that includes data from northern Alberta SAGD operations. The performance of the approach was discussed in two stages: i) the performance of the one-step LSTM models, which includes feature importance and robustness test, and ii) the performance of the actual recursive approach in terms of R2 and the evolution of the residuals as the prediction horizon increases. Additionally, a genetic algorithm was used to optimize steam injection rates.

Results show that the recursive approach is able to properly forecast reward (net present value) very well when considering daily (Case 1) dynamics as opposed to when considering weekly time steps (Case 2). Results suggest that this is because the reward model for Case 1, depends mostly on non-predicted or given values, such as previous steam injection rates and cumulative steam injection. In Case 2, the prediction of reward, depends significantly on predicted values, like, oil production rates for well 4 and 5, and cumulative oil production. Regarding oil production prediction, in both cases, these values depend greatly on previous values of themselves, and moreover, small variations of these values have a significant impact on the prediction values. As a consequence, it was found that the residuals of the recursive prediction of reward for Case 1, don't grow as much as the oil rates prediction.

Chapter 6

A corrected recursive based dynamic surrogate model and optimization approach[1]

6.1 Introduction

In this chapter, we propose a new method for the prediction of time-varying outputs of a dynamical system that can be used to solve the optimization problem expressed in Equation 2.1. This method consists of modeling the residuals exhibited by the conventional recursive-based prediction approach (Chapter 5) and then using this value as a correction term. It relies on the assumption the residuals of the recursive approach are correlated with time and thus can be generalized over the input space.

Results show that the corrected model significantly outperforms the conventional recursive approach by, reducing error (MAPE) and increasing fit (R2). Furthermore the method shows promise as a general framework for effective and efficient identification of dynamic surrogate models.

[1]A version of this chapter will be presented in the 2023 SPE Canadian Energy Technology Conference and Exhibition to be held on March 14-16 in Calgary,Alberta, Canada

6.2 Motivation

Conventional recursive base prediction consists of identifying one-step forecast model(s) - $\hat{\mathbf{g}}(x_t, u_t)$ - of the dynamics of the system, *i.e.*

$$x_{t+1} = \mathbf{g}(x_t, u_t) \approx \hat{\mathbf{g}}(x_t, u_t) + \epsilon \tag{6.1}$$

where x_t and u_t represent the state of the system and the control action at time t, respectively, and ϵ is the prediction error.

Afterward, n-step prediction is achieved by using Equation 6.1 recursively. For instance, at time $t = 1$, x_1 and u_1 are used as inputs, and x_2 is calculated. Then, with a new control action, u_2 along with the predicted x_2, x_3 is calculated. This process is repeated until reaching the desired step number (n).

By inspection of Equation 6.1 we can see that after every prediction step there is an accumulation of the error term ϵ. As a consequence, as the number of time steps increases, so will the prediction error; the errors are propagated. Furthermore, as stated by [108], these errors may exhibit an exponential form as n increases.

6.3 Proposed Methodology

To overcome the situation described in Section 6.2 we propose the use of a correction term that can predict the error term exhibited by the recursive approach (base case). In other words, we assume that errors are a function of x_t and u_t. Thus, the goal is to find a model ϵ such that,

$$\epsilon_t = \epsilon(x_t, u_t) \tag{6.2}$$

Then Equation 6.1 can be modified as,

$$x_{t+1} = \mathbf{g}(x_t, u_t) \approx \hat{\mathbf{g}}(x_t, u_t) + \epsilon(x_t, u_t) \tag{6.3}$$

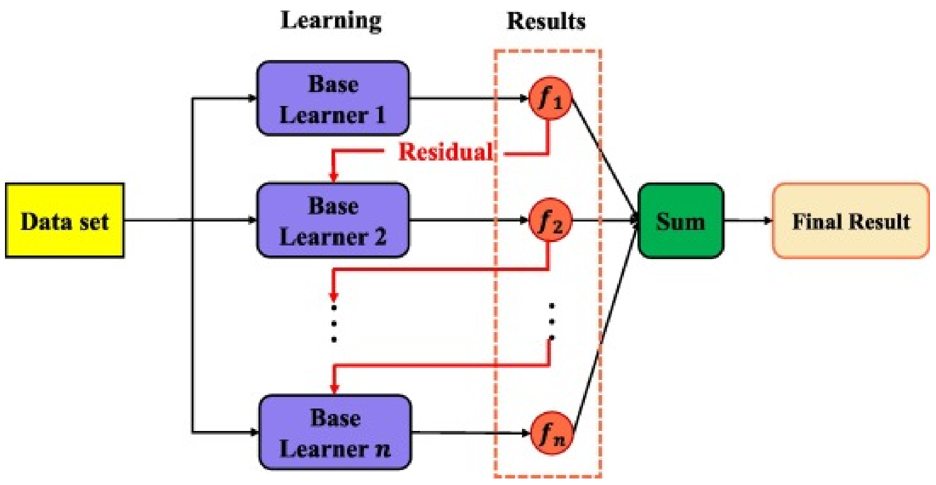


Figure 6.1: Illustration of the gradient boosting framework. Credits: [53]

We can draw two (2) parallelism from the proposed methodology: geostatistical Kriging method and gradient boosting for supervised learning. On one hand, in Kriging, the spatial prediction is decomposed into a trend or mean and a residual term that is correlated with distance from the known values. In a similar fashion, in this work we propose (Equation 6.3) to decouple the conventional recursive prediction given by $\hat{\mathbf{g}}(x_t, u_t)$ and the error term $\epsilon(x_t, u_t)$. However, unlike Kriging, here, we assume residuals are correlated with time from the known value or, in this case, the initial value.

On the other hand, gradient boosting [26], [27] consists of sequentially identifying an ensemble of weak learners where in each stage, each learner will compensate for the weakness of existing learners (Figure 6.1). In this context, a weak learner is defined as a model that is able to perform slightly better than random chance. So, initially, the first learner $\hat{f}(x)$ is identified using the training data by minimizing a predetermined loss function; $\hat{f}(x)$ will map $x \rightarrow y$. Then, the residuals h can be estimated as,

$$h_1(x) = y - \hat{f}(x) \tag{6.4}$$

Then, a second model is identified using x as input and h as output; $\hat{h}(x)$ will map $x \to h_1$. Now, the prediction is corrected as,

$$\hat{y} = \hat{f}(x) + h_1(x) \tag{6.5}$$

In general, for M weak learners, the prediction can be expressed as,

$$\hat{y}_M = \sum_{m=1}^{M} h_m(x) \tag{6.6}$$

where h_m is defined as the residuals resulting from the prediction made by the previous learner(s), *i.e.*

$$h_m(x) = y - \hat{f}(x) + \sum_{i=1}^{m-1} h_i(x) \tag{6.7}$$

Using Equation 6.3 we can now describe the four (4) steps of the methodology, *i.e.* design of experiments, evaluation of each sample, surrogate model ($\hat{\mathbf{g}}(x_t, u_t)$,$\epsilon(x_t, u_t)$) identification and validation, and optimization.

6.3.1 Design of experiments (DOE) - input

Involves generating a sampling plan over the entire design variable space. In particular, implies selecting k steam injection policies for each of the m considered well pairs. Where a steam injection policy is defined as the steam injection rate at every time step for a particular well pair (Figure 5.2).

The primary interest of any sampling plan is to minimize bias and reduce variance [78]. Where *bias* measures the difference between the estimations of the surrogate model and the true values, *e.g.* mean square error, and *variance* quantifies the sensitivity of the surrogate model to a particular sample or data set. This arises to the well-known bias-variance trade off, in which surrogate models that exhibit a low bias to a particular data set will typically show a large variance.

Although increasing the number of samples or in this work, the number of steam injection policies, will reduce both the bias and the variance, this will imply an increasing model complexity. Furthermore, the number of samples is heavily constraint by the computational expense required to evaluate each sample. Alternatively, DOEs that can sample points uniformly have also been proven to reduce bias and variance. In this context, uniformly can be defined by, for example, maximizing the minimum distances between each sample in the set. To this end, practical implementations include Latin Hypercube and optimal LHS, however, in most cases, these methods are designed for time independent variables.

In the field of system identification, where variables are time dependent, common DOE or input signals are Filtered Gaussian White Noise and Random Binary Signal. In this context, these signals, although typically constrained by minimum and maximum values, could exhibit drastic behaviors that are unrealistic in the context of SAGD operations.

In this work, we propose the generation of steam injection policies by having the value at time $t+1$ dependent on the value at time t by three (3) possible *actions*. For example, starting at a specific value of for example $u_0 = 200\text{m}^3/\text{d}$, then u_1 is given by one of three actions: either adding/subtracting a constant value or no change with respect to u_0. Furthermore each action is chosen randomly and both bias and variance (as described above) are evaluated. This evaluation is made on the test set that results by dividing the data in the conventional train-test split.

6.3.2 Evaluation of each sample using high-fidelity model - output

The samples generated by the DOE are used as input to a numerical reservoir simulation model. This is a first principle model that consists of a set of partial differential equations built from, material balance equations (one for each hydrocarbon component), energy balance equation, thermodynamic phase equilibrium conditions, and constraint equations (sum of saturations, oil-phase mole fraction and gas-phase mole fractions must be 1).

Additionally, two (2) types of border conditions are used:

- Dirichlet condition. Bottom hole pressure specified in producing wells
- Neumann condition. Used to specify no-flux (mass and heat) boundaries (at the limits of the domain or reservoir) and steam injection rates at a given temperature in the injecting wells

Regarding initial conditions, initial saturation is determined by irreducible water saturation given by the relative permeability curves. In this case, water-oil and gas-oil contacts are considered well below and above the oil reservoir, respectively. Initial pressure and temperature are estimated from the value of pressure and temperature at a reference depth (datum); pressure is then calculated using the condition of hydrostatic equilibrium and temperature is considered constant.

Each of the samples generated in the DOE will represent a different numerical reservoir simulation model. Thus, the solution for each of these models will result in two (2) time-varying outputs of interest for each considered well pair *i.e.* water (q_w) and oil (q_o) production rates for each considered well pair.

6.3.3 Surrogate model identification and validation

To predict oil q_o and q_w two functions - $\hat{\mathbf{g}}(x_t, u_t)$,$\epsilon(x_t, u_t)$ are identified for each considered well pair (a total of four models) following:

For q_i in $[q_o, q_w]$,

1. Define x_t by including the p previous values of steam injection rates q_s, q_i and the time step number
2. Identify a one-step forecast model $\hat{\mathbf{g}}(x_t, u_t)$
    2.1. Evaluate $\hat{\mathbf{g}}(x_t, u_t)$ using the test data
3. Calculate the residuals exhibited by $\hat{\mathbf{g}}(x_t, u_t)$ by using a conventional recursive approach on the train data
4. Identify the correction model $\epsilon(x_t, u_t)$ using the residuals from Step 3
    4.1. Evaluate $\epsilon(x_t, u_t)$ using the test data
5. Evaluate the bias of the model by using Equation 6.3 with the test data

To evaluate the variance or the effect of randomness over the DOE, these steps are repeated for several different statistical realizations.

6.3.4 Optimization

The well-known Net Present Value (NPV) global economic objective function is used, *i.e.*

$$J = \sum_{t=1}^{T} \frac{P_o \sum_{j=1}^{n} q_{o,j,t} - C_{steam} \sum_{j=1}^{n} q_{s,j,t} - C_{water} \sum_{j=1}^{n} q_{w,j,t}}{1 + i^{\frac{t-t_{ref}}{365}}} \tag{6.8}$$

where, P_o is the oil price [USD/STB], $q_{o,j,t}$ the cumulative oil production obtained in time step t for well pair j [STB], C_{steam} the cost of steam generation [USD/STB], $q_{s,j,t}$ the cumulative steam injection applied in time step t for well pair j [STB], C_{water} the cost of produced water handling [USD/STB], $q_{w,j,t}$ the water production observed in time step t for well pair j, i is the annual discount factor [fraction], T is total production horizon and t_{ref} is the reference time to which NPV is discounted.

Regarding the design variables, as described in Section 6.3.1, these are represented by one discrete number $\{0, 1, 2\}$ per considered well pair, thus, any integer optimization algorithm can be used. This representation is very useful for optimization purposes, however the u_t value used to identify the surrogate models is represented by the real steam injection value.

6.4 Case studies

The described methodology is applied to three (3) case studies with a varying number of considered well pairs, *i.e.* 1, 3, and 9. For the rest of this work these case studies will be referred to as, CS1, CS3, and CS9. In this section, the parameters for these case studies are described according to the steps described above.

6.4.1 Design of experiments - input (case studies)

For CS1, two (2) sample sizes are tested to evaluate variance (as defined in Section 6.3.1): 100 (reduce sample) and 500 (extended sample). For CS3 and CS9 only the reduced sample will be considered.

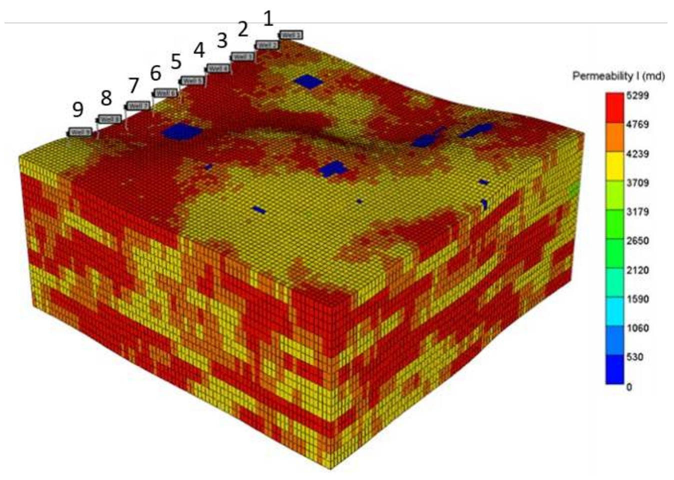


Figure 6.2: 3D grid and porosity field (case studies)

6.4.2 Evaluation of each sample using a high fidelity model - output (case studies)

The numerical reservoir simulation model (high fidelity) used is the same one described in Chapter 5. This model was built using publicly available data of heavy oil SAGD projects CMG-STARS training examples. In particular, petrophysical properties were derived from data from the Christina Lake Thermal Project [12] in Canada and thermal properties, fluid components, and rock-fluid data were taken from heavy oil models available in CMG-STARS [19]. The model consists of nine well-pairs (Figure 6.2) which are opened according to the respective case study. For instance, CS1 only well 5 (middle well) is open for injection and production, while the rest of the wells are shut. For CS3, wells 4, 5, and 6 are used and for CS9 they are all used.

Table 6.1 shows the general characteristics of the reservoir model. For the rest of the parameters, such as, grid size, thermal properties, fluid component definition, oil viscosity changes with respect to temperature, relative permeability curves, the reader is referred to Chapter 5, Section 4.

Table 6.1: Reservoir simulation model parameters (case studies)

Reservoir data	Reservoir compressibility [kPa^{-1}]	1.00×10^{-6}
	Rock heat capacity [J/m^3°C]	2.3×10^6
	Rock thermal conductivity [J/m day°C]	2.7×10^5
	Reservoir initial temperature [°C]	48.4
	Reservoir initial pressure [kPa]	2011
	Initial water saturation	0.19
	Initial oil saturation	0.74
	Average porosity	0.29
	Average permeability [mD]	4275
	Well spacing [m]	80
Production parameters	Production pressure [kPa]	1000
Production horizon/episode length [d]		250

6.4.3 Surrogate model identification and validation (case studies)

Section 6.4.1 and 6.4.2 describe the procedure to generate the input and output data, respectively. In particular, we define the input data x_t as, the 40 previous values of steam injection rates and oil/water production rates, and the current steam injection rate value to be applied at time t. Thus, for both, $\hat{\mathbf{g}}(x_t, u_t)$ and $\epsilon(x_t, u_t)$ the input vector will be of size 81, and the output vector of size 1. The following sections describe the identification and validation strategies for all models.

Identification

For all case studies, Long-Term Short Memory (LSTM) were used to identify $\hat{\mathbf{g}}(x_t, u_t)$ and $\epsilon(x_t, u_t)$. LSTM are non-linear neural networks particularly designed to capture time series or sequenced data. In particular, the goal is to use previous information (long/short term) to be able to make predictions for the near future. This is achieved by introducing the concept of cell state and three(3) gates (Figure 6.3), these are:

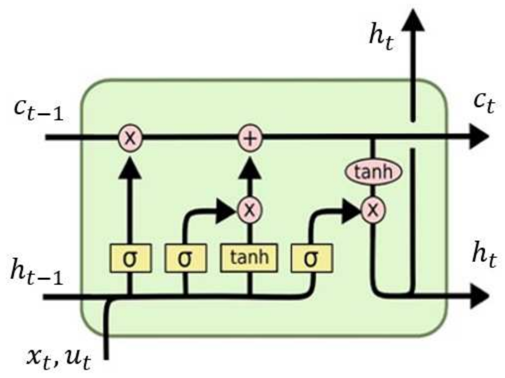


Figure 6.3: LSTM architecture. Figure credits: [74]

forget, input and output. In this sense, the LSTM is designed with a conveyor belt dynamic, in that information is removed or added to the so-called cell state c_t by the gates. A high-level description of the gates follows as:

1. **Forget gate**. Here, the decision is made on what information is to be forgotten or removed from the cell state. The input is the hidden state from the previous h_{t-1} and the input data x_t.

   This implies the application of the sigmoid function:

$$f_t = \sigma(W_f \cdot [h_{t-1}, [x_t, u_t]] + b_f) \tag{6.9}$$

   As a result of this operation, f_t will represent a vector of 0s and 1s, *i.e.* information that is forgotten and kept, respectively. Afterwards the multiplication operator is applied to f_t and c_{t-1}.

2. **Input gate**. The next step to add new information to the cell state. This is done by applying again the sigmoid (Equation 6.9) and tanh function to f_t and c_{t-1}, *i.e.*

$$\begin{aligned} i_t &= \sigma(W_i \cdot [h_{t-1}, [x_t, u_t]] + b_i) \\ \tilde{c_t} &= \tanh(W_c \cdot [h_{t-1}, [x_t, u_t]] + b_c) \end{aligned} \tag{6.10}$$

Then, i_t and $\tilde{c_t}$ are multiplied and added to the current cell state (Equation 6.10), as,

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c_t} \tag{6.11}$$

The first part of Equation 6.11 results again in vectors of 0s and 1s, and the second part represents the degree to which the information is kept is updated.

3. **Output gate**. This gate decides on the actual output values h_t that will consist of a filtered version of the current cell state. Mathematically,

$$\begin{aligned} o_t &= \sigma(W_o \cdot [h_{t-1}, [x_t, u_t]] + b_o) \\ h_t &= o_t \cdot \tanh c_t \end{aligned} \tag{6.12}$$

   Notice in Equation 6.12 h_t consists of two parts; the first corresponds to once again a sigmoid function applied to the input. This is multiplied by the tanh function applied to the current cell state. Intuitively, this means that the first part decides which elements of the cell state will correspond to the output, and the second part decides to what extent the current cell state is modified.

The LSTM architecture and the corresponding parameters used to model $\hat{\mathbf{g}}(x_t, u_t)$ and $\epsilon(x_t, u_t)$ are shown in Table 6.2.

Validation

This can be separated in two stages, validation of the one-step forecast and residual models, and validation of the long term prediction capacity of the framework. In the first stage the training process of the LSTM models are tested using $R2$, *i.e.*

Table 6.2: Parameters used to identify $\hat{\mathbf{g}}(x_t, u_t)$ and $\epsilon(x_t, u_t)$

Parameter	$\hat{\mathbf{g}}(x_t, u_t)$	$\epsilon(x_t, u_t)$
Number of layers	2	4
Number of neurons per layer	80	280
Interleaved dropout layers rate	0.2	
Optimizer	Adam	
Learning rate	1×10^{-3}	1×10^{-4}
Batch size	50	100
Train-test split ratio	80-20	

$$R2_k(y, \hat{y}) = 1 - \frac{\sum_{t=1}^{T} (q_{k,t} - \hat{q}_{k,t})^2}{\sum_{t=1}^{T} (q_{k,t} - \bar{q}_{k,t})^2} \tag{6.13}$$

where $q_{k,t}$ represents the true production rate values for either water or oil, $k = \{o, w\}$ at time t as obtained via the numerical reservoir simulation model (Section 6.4.2), $\hat{q}_{k,t}$ is the prediction made by the LSTM neural networks, $\bar{q}_{k,t}$ the arithmetic average of the true values, and T is the prediction horizon.

The second phase, is the evaluation of the n-step prediction capacity of the steps described in Section 6.3.3. Here we use two performance measures, $R2$ and Mean Absolute Percentage Error (MAPE). $R2$ is defined similarly as in Equation 6.13 except that $\hat{q}_{k,t}$ is the prediction values given by the corrected or base model methodology described before. Additionally, MAPE is given by,

$$MAPE_k(y, \hat{y}) = \frac{1}{T} \sum_{t=1}^{T-1} \frac{|\mathbf{q}_{k,t} - \hat{\mathbf{q}}_{k,t}|}{\max(\eta, |\mathbf{q}_{k,t}|)} \tag{6.14}$$

where $\mathbf{q}_{k,t}$ represents a vector that contains the true production rate values for all considered well pairs, either water or oil, $k = \{o, w\}$ at time t as obtained via the numerical reservoir simulation model, $\hat{\mathbf{q}}_{k,t}$ is the prediction values offered by the

Table 6.3: Economical parameters (case studies)

Parameter	Value
P_o [USD/STB]	60
C_{steam} [USD/STB]	10
C_{water} [USD/STB]	5
i [fraction]	0.1
t_{ref} [day]	0

corrected or base case methodology described above. Additionally, η, as described in [76], is an arbitrary small and positive number to avoid undefined results when $\mathbf{q}_{k,t}$ is zero, and T is the number of samples.

6.4.4 Optimization (case studies)

In terms of the objective function described by Equation 6.8, the economical parameters are shown in Table 6.3. Furthermore, a Python based implementation [83] of a genetic algorithm was used to solved the integer optimization problem described in Section 6.3.4.

Genetic algorithms are non-gradient optimization algorithms in which initial solutions are evolved to optimal solutions by biologically inspired operators, *e.g.* crossover, mutation. In this context, solutions or individuals are steam injection policies for all considered well pairs represented by integers or actions. In general the algorithm consists of three (3) stages, initialization, creating a next generation and termination.

Initialization refers to starting with an initial population or a predetermined number of candidate solutions, typically generated randomly. Each of these candidate solutions are evaluated using the dynamic surrogate models and the solutions are ranked according to their corresponding NPV value or performance. The population is referred to as a *generation* and as the iterative process continues new generations are generated with higher performance.

In the second stage, the new generation is created in three (3) ways,

- Elitism. Best individuals are selected to continue on to the following generation
- Crossover. These individuals are a result of the combination of two (2) good performing parents. The offspring will share characteristics of the parents and will tend to have a better performance or NPV value.
- Mutation. Implements random changes to selected individuals

These two stages are repeated until a termination condition is achieved and the algorithm is terminated (third stage). These conditions may include, evolving for a pre-determined number of generations or iterations and the NPV values of the best-performing individuals do not differ significantly from generation to generation.

6.5 Results and discussion

In this section, the results for all three (3) - CS1, CS3, CS9 - case studies are presented on the basis of the validation phases described in Section 6.3.3. The first phase corresponds to the evaluation of the one-step forecast capacity of the models. Additionally, an assessment is made on how the residuals change as the number of prediction steps n increases. The second phase consists of evaluating the performance of the residual model $\epsilon(x_t, u_t)$ individually and when coupled with the base case $\hat{\mathbf{g}}(x_t, u_t)$.

6.5.1 CS1 - One well

First phase

The performance of the one-step forecast models $\hat{\mathbf{g}}(x_t, u_t)$ for oil and gas are shown in Figure 6.4. These models correspond to following Step 2 and 2.1 of the steps

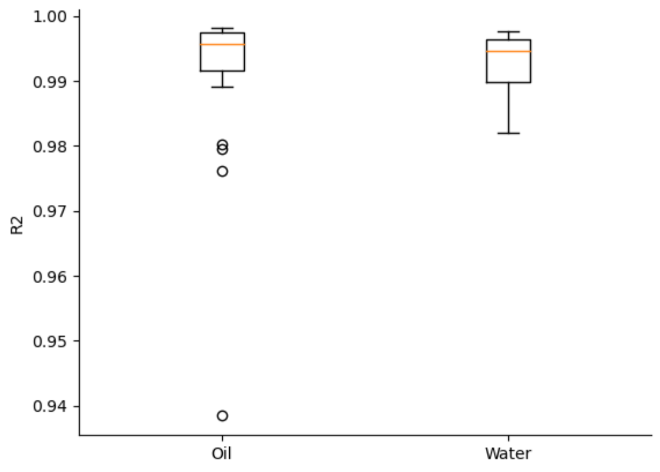


Figure 6.4: Distribution of R2 values exhibited by the oil (left) and water (right) one-step forecast model using the test data of the limited sample size

described in Section **??**. Here we see the distribution of the R2 values estimated using the test data of the limited sample size. More specifically, each value in these boxplot results in comparing the predicted scaled oil/water rate values and comparing with the true values as obtained via numerical simulation (Figure 6.5). Note that both models perform very well giving R2 values are well over 0.94 in all cases.

This one-step forecast model is now used recursively for n-step prediction (base case), and the residuals exhibited are shown in Figure 6.6. Here we show the average and one standard deviation of the residuals exhibited by $\hat{\mathbf{g}}(x_t, u_t)$ as the number of prediction steps n increases for the test data using the limited sample size. We can see how for oil prediction, residuals significantly increase as n increases, confirming some level of correlation and thus justifying the use of the residual model described in Section 6.3.3. However, for water prediction, the residuals are notably lower. For example, water prediction residuals reach a maximum value of around 0.2 while oil prediction residuals maximum value is more than 1.5; almost 7 times more.

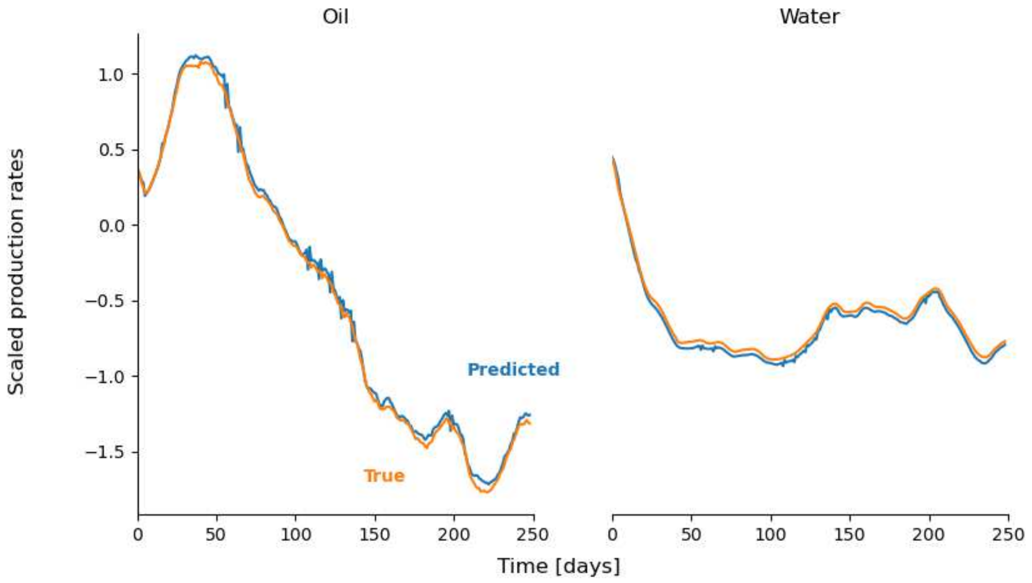


Figure 6.5: Prediction values of scaled production rates (oil and water) compared to true values for one representative sample

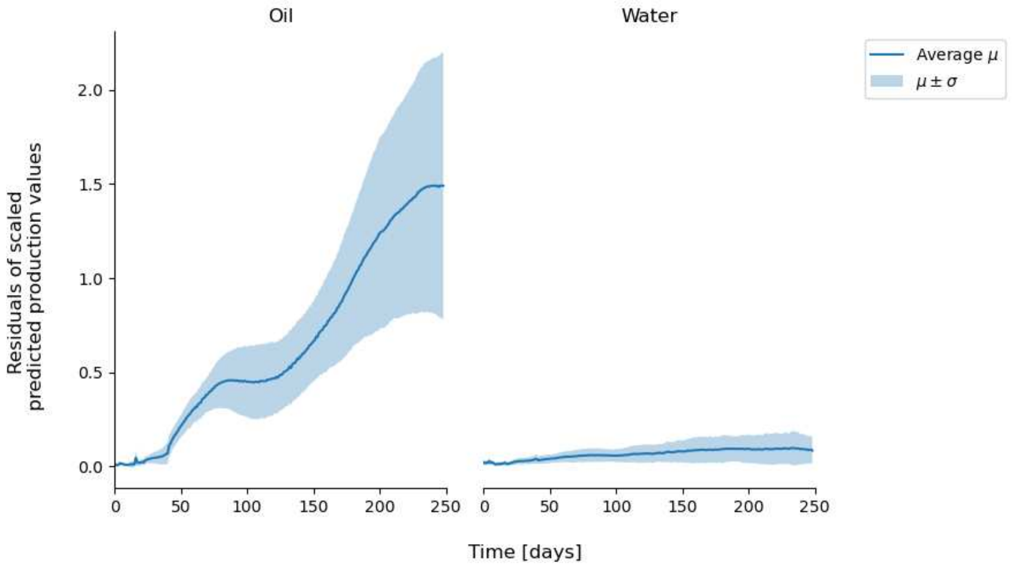


Figure 6.6: Average and one standard deviation of the residuals for the oil (left) and water (right) model as the number of prediction steps n increases for the test data using the limited sample size

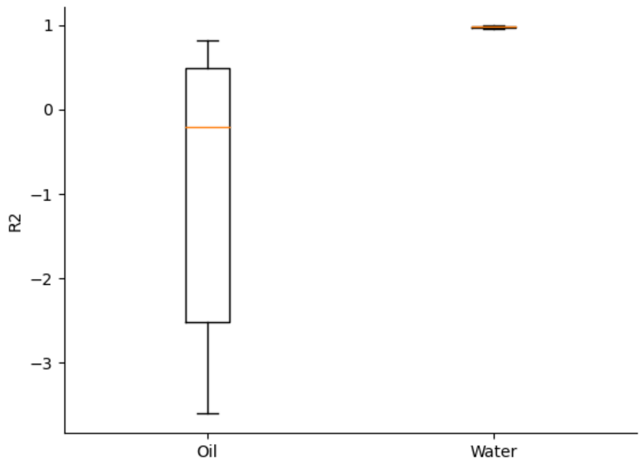


Figure 6.7: Performance of the conventional recursive approach for oil and water prediction using the test data

The results in Figure 6.6 are confirmed by looking at the performance of the conventional recursive model in terms of R2 as shown in Figure 6.7. This figure shows the distribution (without outliers) of R2 values for the prediction of oil and water rates. Note how oil prediction R2 values are in general very low, with a very high variability, from -7.6 to 0.81, and with a median of -0.21. Compared to water prediction R2 values that range from 0.95 to 0.99, with a median of 0.98.

One possible explanation for such positive results for water prediction is the correlation between steam injection policies and water production values. Figure 6.8 shows the distribution of the correlation coefficient between the steam injection rate values (input) and oil/water production rate values (output) for all the samples in the test data. We can see how water production rates are significantly more correlated with the steam injection rates, as compared to oil production rates. In particular, water production rates show a median correlation value of 0.76, while oil production rates have a median correlation of -0.1.

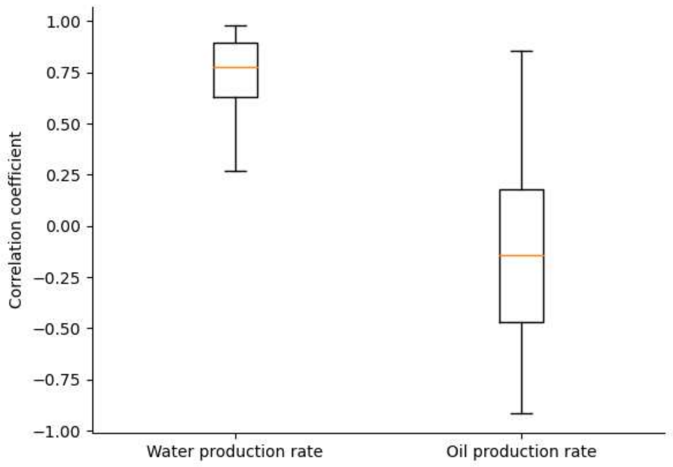


Figure 6.8: Correlation values between steam injection policies (input) and oil/water production rates (output)

Furthermore, we can also note significant differences between the oil and water models when performing a feature importance and robustness test on each of the models. The feature importance test corresponds to the implementation available in [76]; here, a base performance measure is estimated $R2_{base}$ using the test data. Then, each feature is randomly shuffled to generate a corrupted version of the test data, and a new $R2_{corrupted}$ value is calculated. This procedure is repeated K times and the importance of each feature j is then estimated as:

$$i_j = R2_{base} - \frac{1}{K}\sum_{k=1}^{K} R2_{corrupted,k,j} \tag{6.15}$$

The robustness test consists of corrupting each feature by adding a noise drawn from a normal distribution of mean zero and varying sigmas. Sigmas are varied from 0 (uncorrupted case) to 0.8 in increments of 0.1, and for each sigma, a perturbed value of $R2$ is calculated.

The results of these tests are available in Figure 6.9, note how for the water rate prediction model (top part of the Figure) the features that have the most impact on the output are, steam rate,$t-3$, water rate,$t-3$, water rate,$t-2$ and steam rate,$t-4$. However, when perturbed, these features have very little impact on the performance of the model where $R2$ is only deteriorated from 0.992 to 0.987. The oil rate prediction model offers different results, on one hand, the most important features correspond to oil rate,t, oil rate,$t-34$, oil rate,$t-33$, oil rate,$t-35$, oil rate,$t-32$, oil rate,$t-31$. Furthermore, these features cause a deterioration of the $R2$ from 0.99 to 0.92. Note how, unlike the water rate prediction model, for the oil rate prediction model the six (6) most important features correspond to predicted values. In the water rate prediction model, these features are a mix of prediction values (water rates) and given values (steam injection rates).

Considering these results, for the rest of the case studies (CS3 and CS9) the residual model will only be applied to the prediction of oil production rates.

Second phase

We begin by evaluating the performance of the residual model $\epsilon(x_t, u_t)$ for oil production rates. As shown in Figure 6.10, this model is able to capture very well the variability of the residuals. In particular, R2 values are well over 0.89 in all cases, with a median value of 0.99.

Coupling the residual model with the one-step forecast model now we can see the performance of the proposed corrected methodology (corrected model) and compare to the conventional recursive approach (base model). Specifically we evaluate the *bias* (as described in Section 6.3.1) of the methodology on the training and test data, as show in Figures 6.11 and 6.12. The figures show the distribution of R2 value exhibited by the base and corrected model using the limited sample size. For example, each sample in the distributions corresponds to comparing the R2 between the predicted time series and the true values; three (3) such comparisons are shown

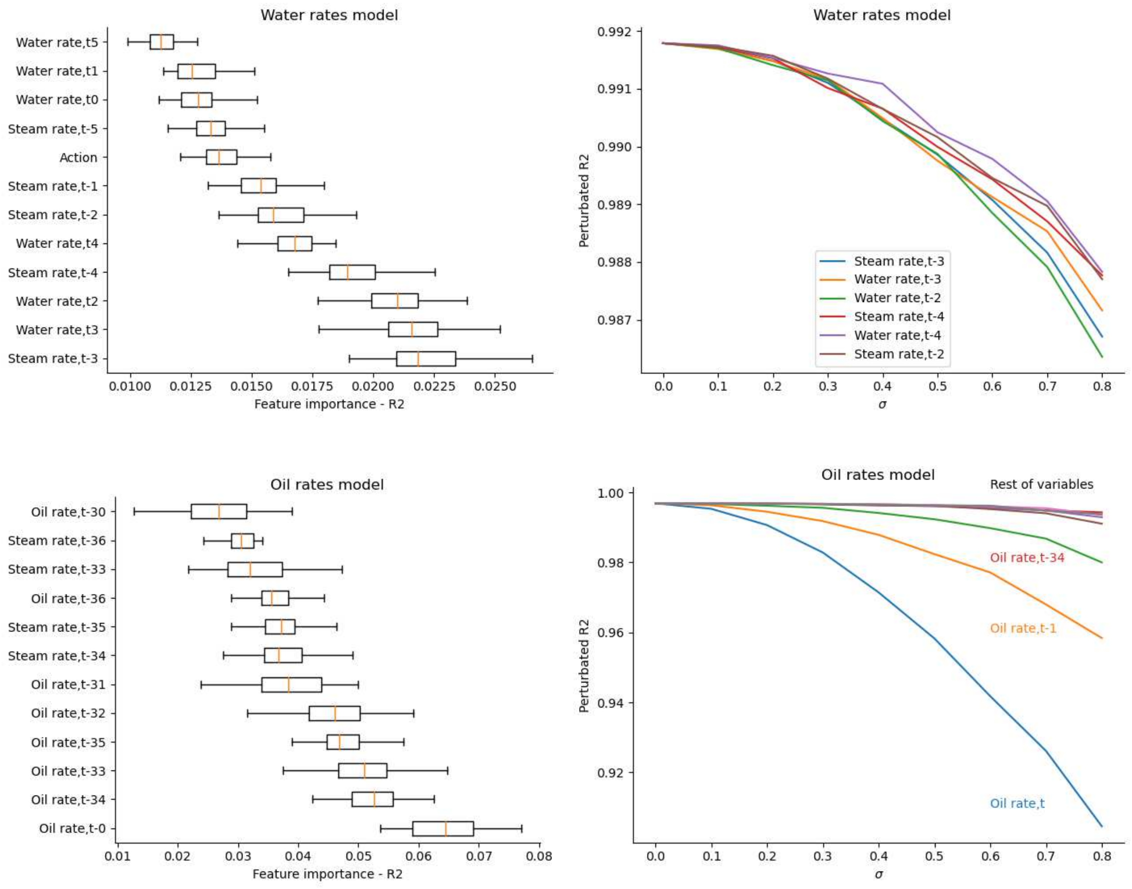


Figure 6.9: Results of the feature importance (left) and robustness (right) tests performed on the water prediction (top) and oil prediction (bottom) rates models

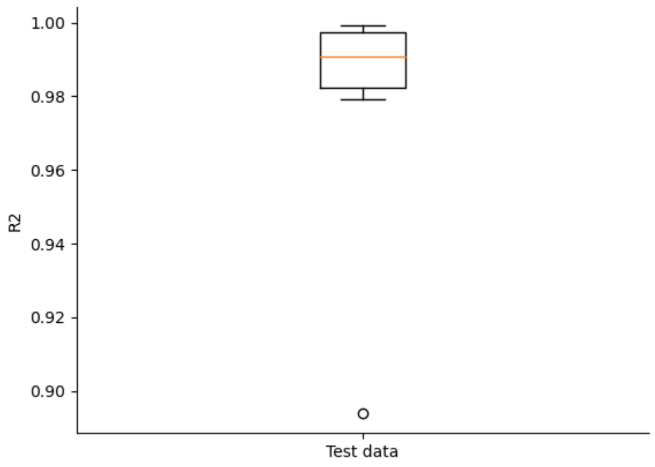


Figure 6.10: Distribution of R2 values exhibited by the corrected model using the test data of the limited sample size

in Figure 6.14.

From Figures 6.11 and 6.12 we can see the significant improvement of R2 values coupling the residual model to make predictions (corrected model) with respect to the conventional recursive approach (base model). On one hand, considering the training data, we see a near perfect prediction. On the other hand, test data also shows very well performance, exhibiting a median of R2 of 0.82, as opposed to the median value for the base case of -0.21. This improvement can also be seen by a significant reduction of the residuals exhibited by the corrected model (Figure 6.13).

To test the *variance* of the proposed methodology, the procedure was repeated for four (4) additional design of experiments (5 in total) considering a R2 and MAPE as performance measures and a limited sample size. Figures 6.15 and 6.16 show the distribution of the corresponding performance measure for five (5) different design of experiments seeds. Note how in all cases, there is a significant increase (decrease) in R2 (MAPE) values.

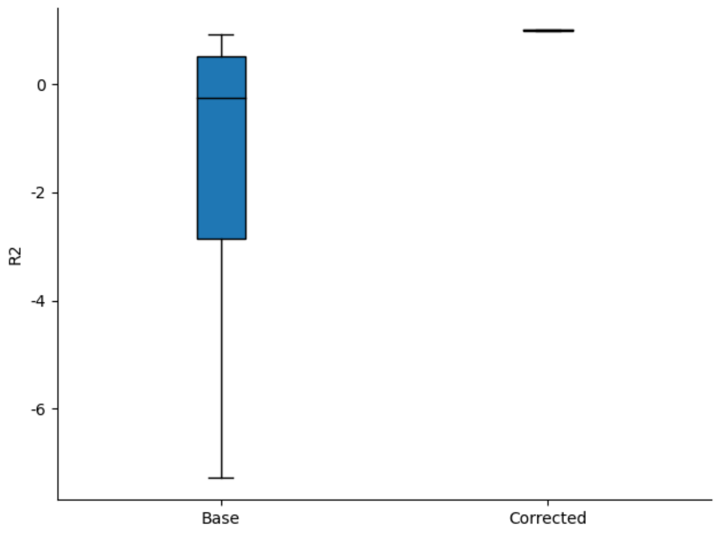


Figure 6.11: Distribution of R2 values exhibited by the corrected and base model using the train data of the limited sample size

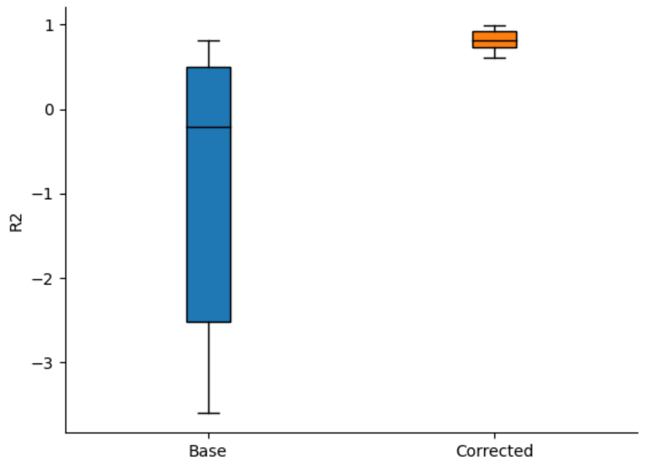


Figure 6.12: Distribution of R2 values exhibited by the corrected and base model using the test data with a limited sample size

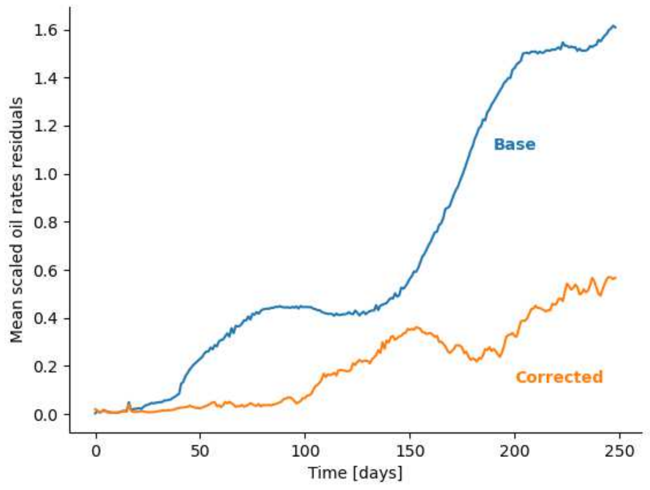


Figure 6.13: Mean residuals exhibited by the base and corrected model

Regarding R2 (Table 6.4), the base model shows median values of between -1.62 to 0.76, while the corrected models shows median R2 values of between 0.83 and 0.96. Table 6.4 also shows the results considering an extended sample size (500), here we see that the increase in sample size shows similar results. In this setting, the median of R2 values for the base model take values between -0.72 and 0.85, while considering the corrected models the range is between 0.85 and 0.99. In general, considering the limited and extended sample size for this well, the global median of R2 for the base model was between 0.62 and 0.64, and the corrected model significantly improved these values to reach a typical R2 of 0.95.

Additionally, as noted in Figure 6.16, for each of the DOEs, the distribution of MAPE values exhibited by the corrected models shows a significantly lower median and standard deviation compared to the base model. In particular, the percentage of this reduction is shown in Table 6.5, notice how for a limited sample size the median is reduced by between 45.9% and 88.6%, while the standard deviation shows reductions of between 38.3 and 83.1%. Using an extended sample size, the median of

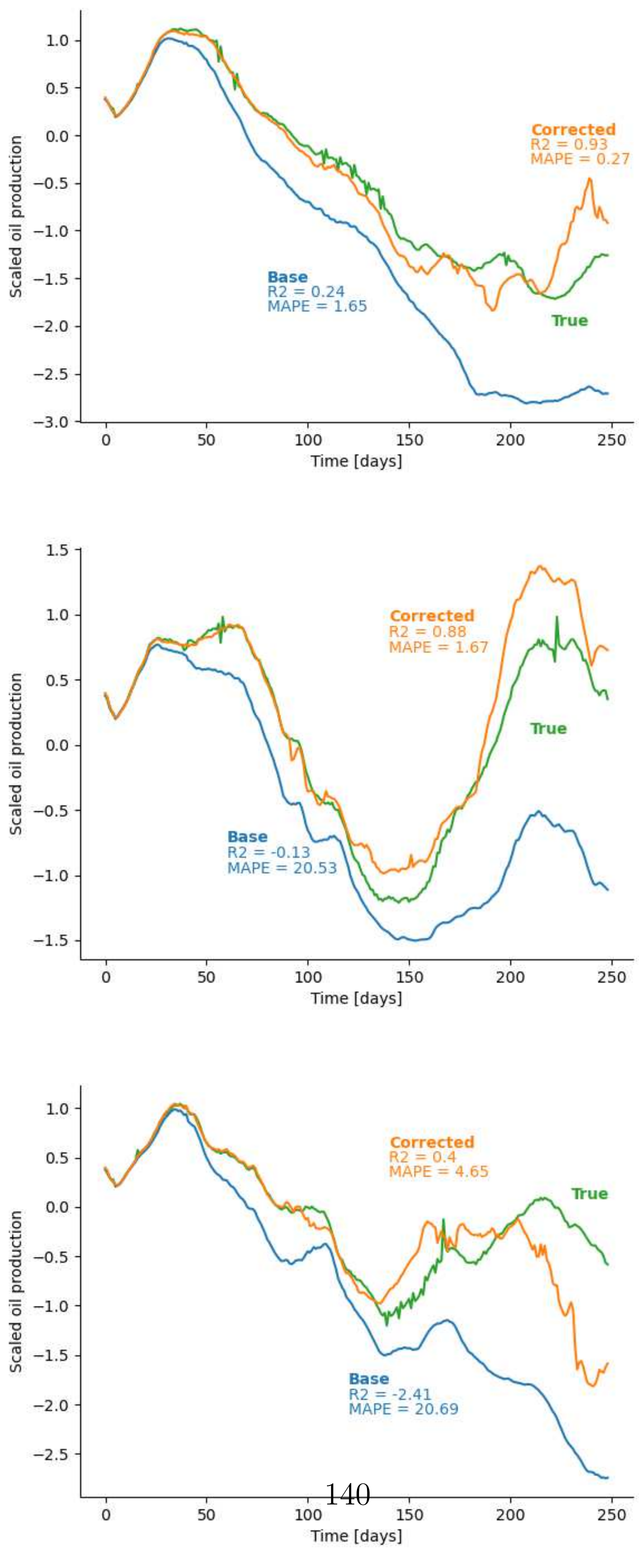


Figure 6.14: Performance of the base and corrected models of three (3) test samples

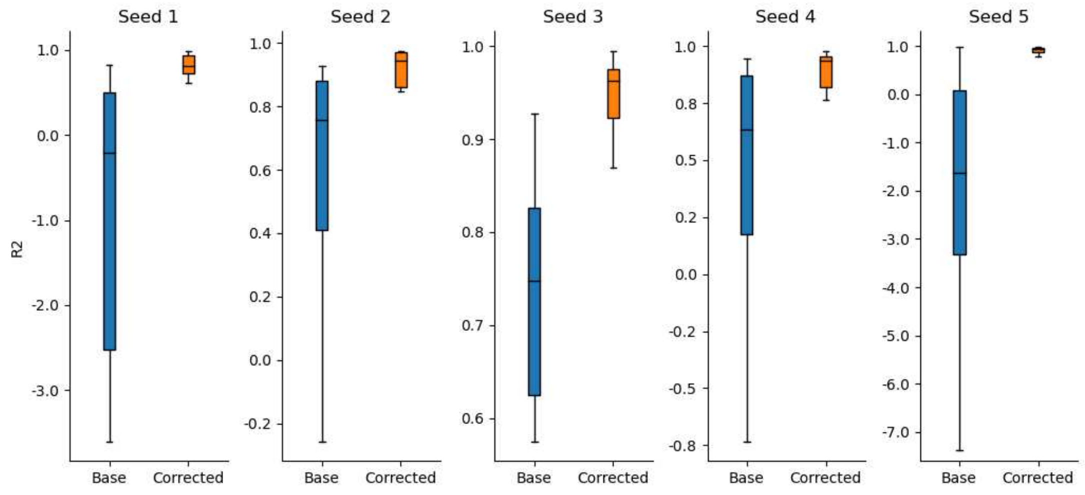


Figure 6.15: Distribution of R2 values exhibited by the corrected and base model considering five (5) design of experiments and using the test data with a limited sample size

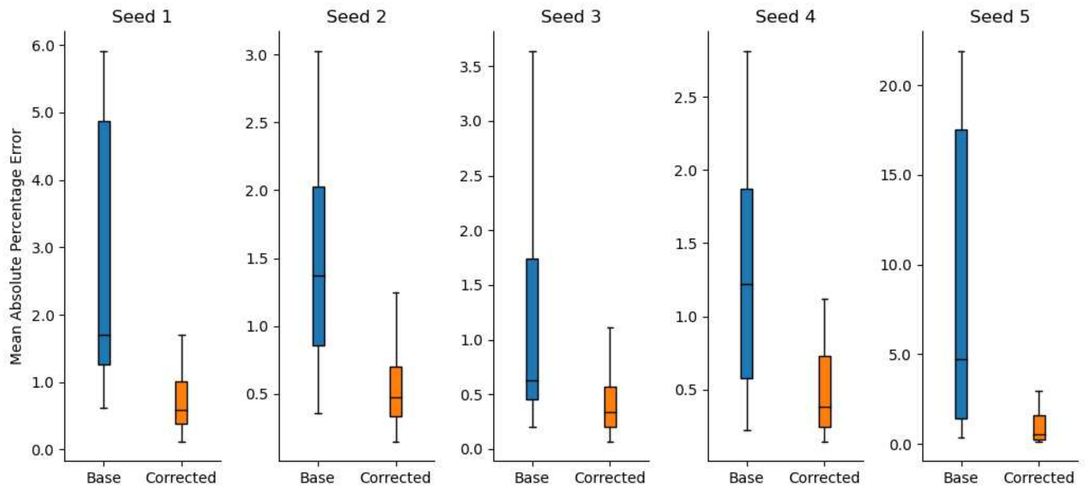


Figure 6.16: Distribution of MAPE values exhibited by the corrected and base model considering five (5) design of experiments and using the test data with a limited sample size

Table 6.4: Median of the R2 distributions exhibited by the corrected and base model considering five (5) design of experiments and using the test data with a limited sample size

DOE	**Limited sample**		**Extended sample**	
	Base	Corrected	Base	Corrected
1	**-0.21**	**0.82**	**0.16**	**0.99**
2	**0.76**	**0.95**	**0.62**	**0.92**
3	**0.75**	**0.96**	**-0.72**	**0.85**
4	**0.64**	**0.94**	**0.78**	**0.98**
5	**-1.62**	**0.95**	**0.85**	**0.95**
Global median	**0.64**	**0.95**	**0.62**	**0.95**

DOE	**Limited Sample**		**Extended sample**	
	Median	**Standard deviation**	**Median**	**Standard deviation**
1	65.8	83.1	82.7	80.1
2	65.4	62.8	53.1	58.8
3	45.9	59.4	84.6	95.1
4	68.7	36.2	84.0	85.2
5	88.6	38.3	55.0	28.6
Global median	67.0	56.0	82.7	80.1

Table 6.5: Percentage of reduction of the corresponding statistic of the MAPE distributions comparing base and corrected model for CS1

the MAPE values is reduced by between 55.0% and 82.7% and the standard deviation is reduced between 28.6% and 95.1%. In general, we can estimate a global median of reduction of median and standard deviation 67.0% and 56.0%, respectively, for the limited sample size, and 82.7% and 80.1% considering an extended sample size. Given the similarity of results obtained with the limited and extended sample size, for CS3 and CS9 only the limited sample size will be use.

Finally, optimization results for a random DOE and a limited sample size are shown in Figure 6.17. Note that steam injection rate is reduced to minimum values until around 100 time steps. Afterwards, it is increased until reaching initial values of

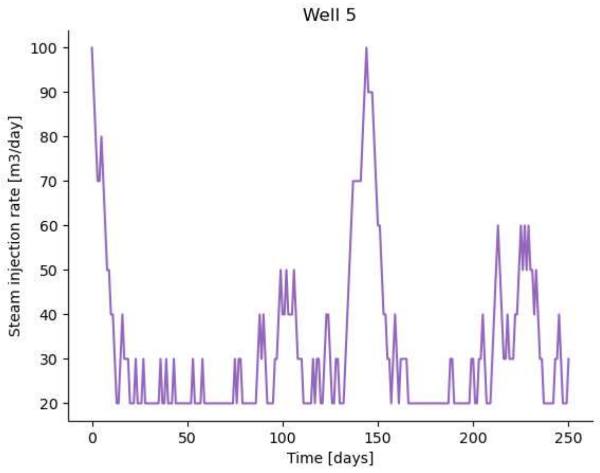


Figure 6.17: Optimal steam injection policy for CS1

100 m3/day, and then immediately dropped to minimum values once again. Then, we note a new increase to around 60 m3/day towards the end of the simulation period.

6.5.2 CS3 - Three wells

Figure 6.18 shows the distribution of the global MAPE (Equation 6.14) values exhibited by the corrected and base model considering five (5) design of experiments. These results correspond to the test data set and a limited sample size, and a representative sample is shown in Figure 6.19. From Figure 6.18 note how the corrected model outperforms the base model by reducing both the median and standard deviation of the MAPE distributions in all cases. In particular, this reduction is shown in Table 6.6, ranging from a 11.6 % to 72.5% reduction for the median, and from 51.0% to 82.8% reduction for the standard deviation. In general, the global median among the DOEs was estimated as 49% reduction of median and 71.2% for standard deviation.

Individual well performance obtained by the base and corrected model is evalu-

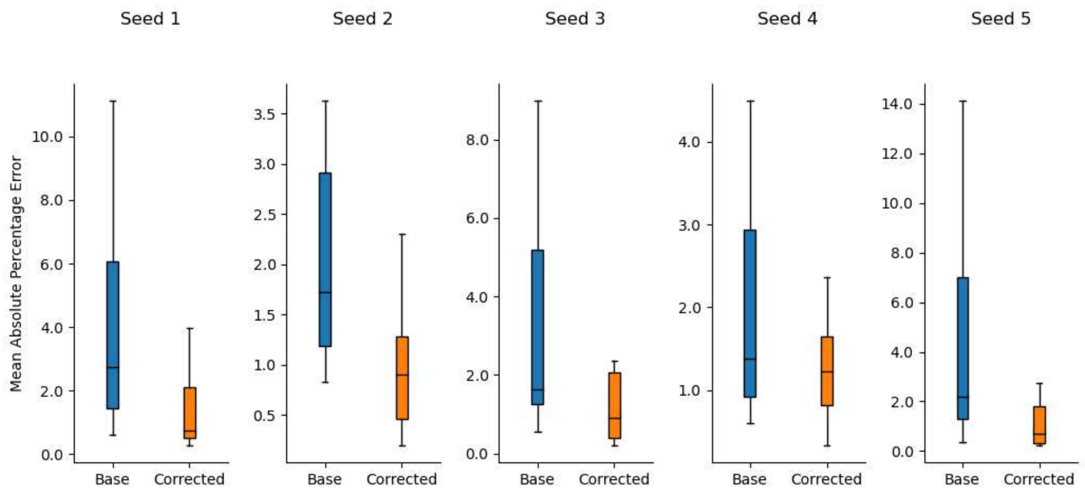


Figure 6.18: Distribution of MAPE values exhibited by the corrected and base model considering five (5) design of experiments of three (3) wells and using the test data with a limited sample size

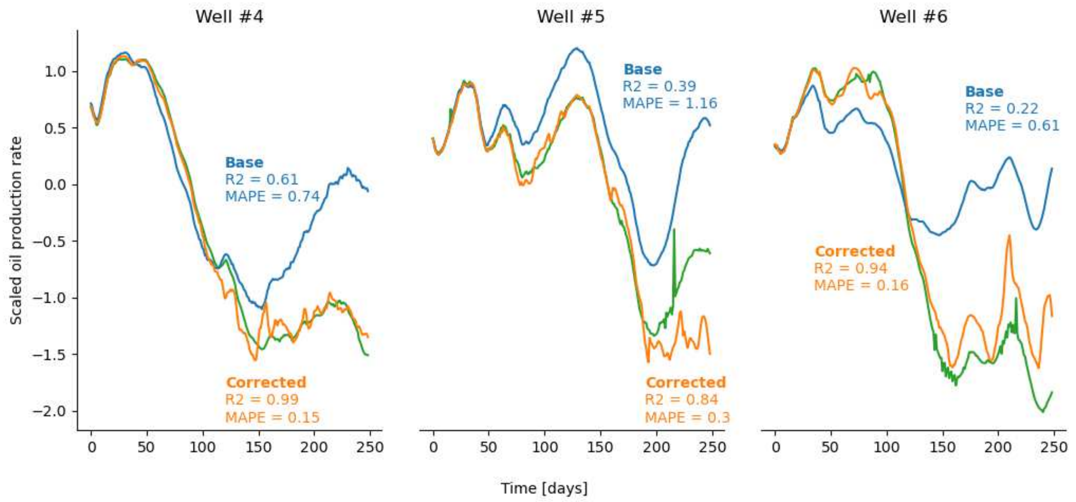


Figure 6.19: Performance of the base and corrected models of a representative three (3) well sample

Table 6.6: Percentage of reduction of the corresponding statistic of the MAPE distributions comparing base and corrected model for three (3) wells

DOE	**Median**	**Standard deviation**
1	72.5	78.4
2	48.0	64.6
3	44.0	79.7
4	11.6	51.0
5	68.0	82.8
Global median	49.0	71.2

ated using R2 as shown in Table 6.7. In all DOEs the corrected model improves R2 significantly for all considered wells. In particular, global median values indicated an increase in R2 value from 0.83 to 0.89 for well 4, 0.72 to 0.92 for well 5 and 0.47 to 0.44 for well 6.

Optimization for a random DOE are presented in Figure 6.20. In all three wells, we can see that the optimal steam injection policy exhibits a sharp decrease of steam injection rates at the beginning of the simulation period. For wells 4 and 5, steam injection rate remains at values of around 40 to 60 m3/day for the most part of the 250 time step production horizon. In well 6, we see that steam injection rates peaks again soon after the sharp decrease at time step 50, only to immediately decrease again, and remain at low values for the rest of the horizon. In Well 5, we also note a sharp increase after time step 200, reaching a value of 120 at the end of the simulation period.

6.5.3 CS9 - Nine wells

Distribution of global MAPE values (Equation 6.14) are shown in Figure 6.21, and Figure 6.22 displays a representative sample of one of the considered DOEs. Similarly to CS1 and CS3, we can note how the distribution of MAPE values exhibited by the corrected model shows a significant reduction of the median and standard deviation,

Table 6.7: R2 values exhibited by base and corrected model for each considered well (4,5 and 6) for five (5) DOEs

DOE	**Model**	**Well**		
		4	5	6
1	Base	**0.92**	**0.89**	**-1.64**
	Corrected	**0.61**	**0.35**	**0.32**
2	Base	**0.61**	**0.35**	**0.32**
	Corrected	**0.86**	**0.82**	**0.88**
3	Base	**0.83**	**0.86**	**0.47**
	Corrected	**0.91**	**0.94**	**0.93**
4	Base	**0.85**	**0.72**	**0.69**
	Corrected	**0.89**	**0.91**	**0.84**
5	Base	**-0.02**	**-0.26**	**0.88**
	Corrected	**0.82**	**0.92**	**0.93**
Global median	Base	**0.83**	**0.72**	**0.47**
	Corrected	**0.89**	**0.92**	**0.88**

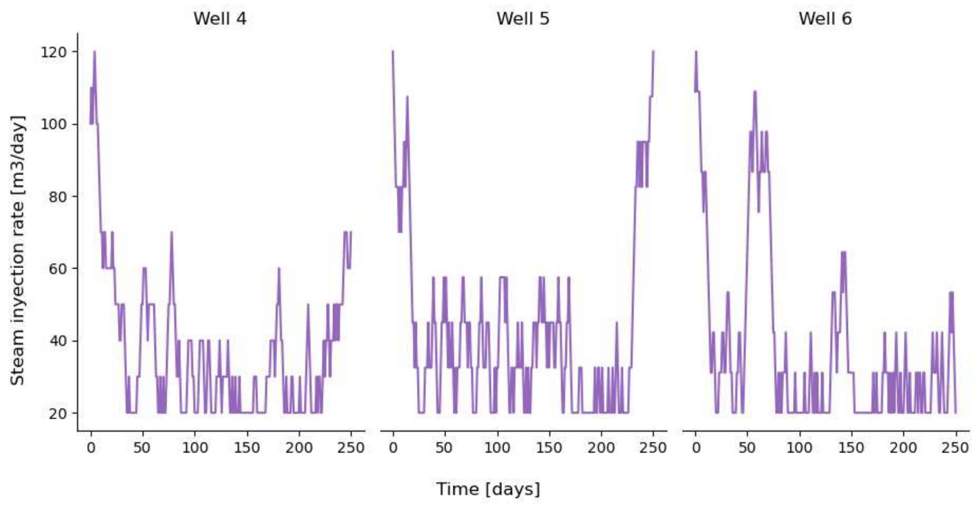


Figure 6.20: Optimal steam injection policy for CS3

Table 6.8: Percentage of reduction of the corresponding statistic of the MAPE distributions comparing base and corrected model for the nine (9) wells case

DOE	**Median**	**Standard deviation**
1	64.6	47.4
2	76.8	65.7
3	60.1	32.9
4	63.6	96.4
5	48.5	65.6
Global median	62.9	61.6

as compared to the base model. In particular, as shown in Table 6.8 this reduction ranges from 48.5% to 76.8% for the median, and between 32.9% and 96.4% for the standard deviation. With global median values of a reduction of 62.9% for the median and 61.6% for the standard deviation.

In terms of individual values, Table 6.9 shows the R2 value exhibited by the base and corrected model for each individual well, for five (5) DOEs. In general, we note a significant increase of R2 for all wells, for all DOEs. This reduction is summarized with a global median value that for the corrected model is over 0.85 for all wells, except for well 1. However, this particular well exhibits a significant improvement over the base model in which the global median value was of -0.55.

The optimal steam injection policies corresponding to the nine well pairs are shown in Figure 6.23. We can see that for some wells, the steam injection rates decrease at the beginning of the production horizon, *e.g.* wells 6, 7, and 9. While for other wells, we see a sharp increase in the rates towards the end of the period of interest, for instance, wells 1, 4, 5. In other wells, we see a sudden increase and decrease in the rate somewhere in the middle of the production horizon, such as is the case for wells 3 and 8.

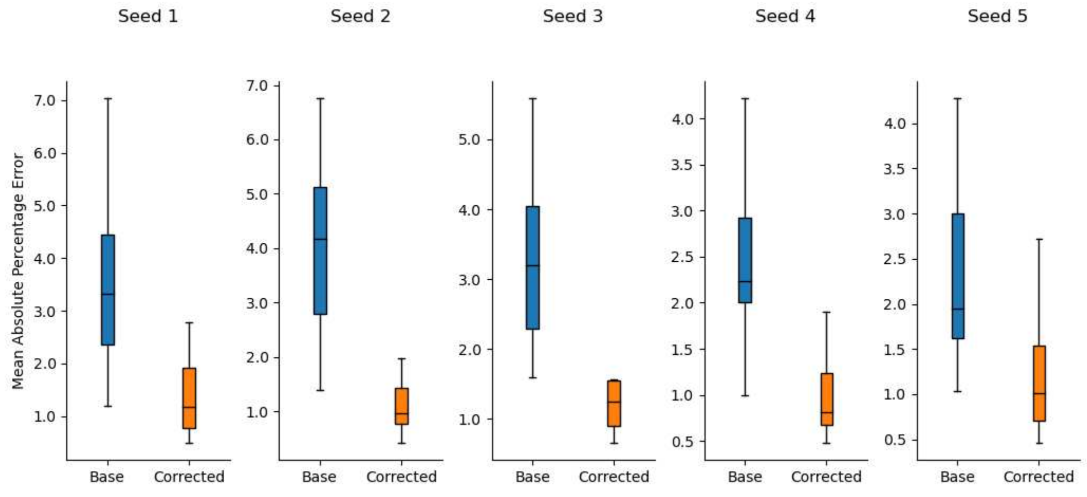


Figure 6.21: Distribution of MAPE values exhibited by the corrected and base model considering five (5) design of experiments of nine (9) wells and using the test data with a limited sample size

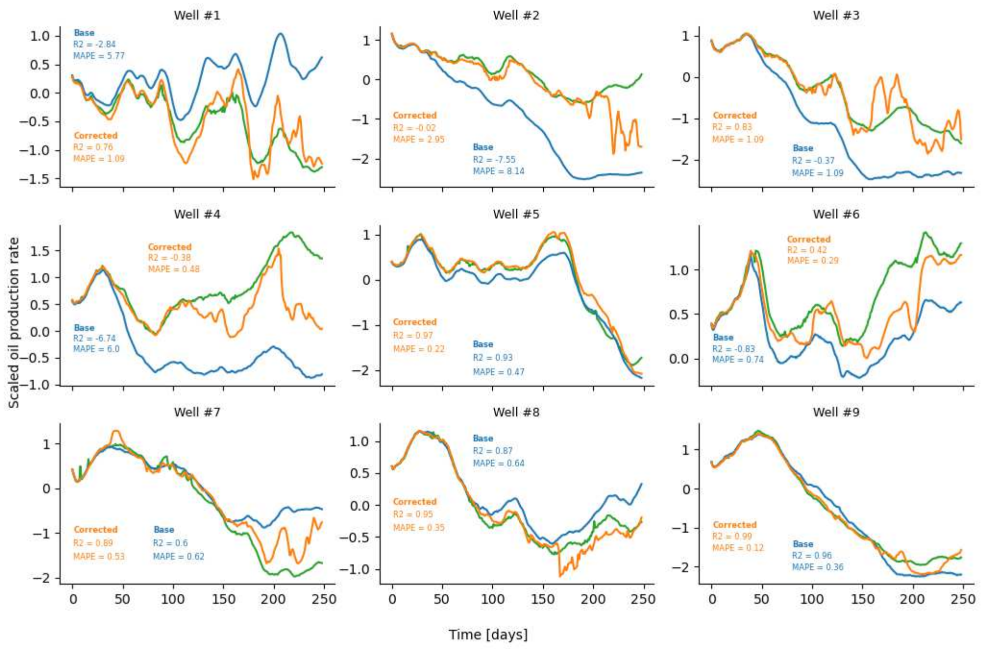


Figure 6.22: Performance of the base and corrected models of a representative nine (9) well sample

DOE	Model	Well								
		1	2	3	4	5	6	7	8	9
1	Base	-1.05	0.13	0.57	0.04	0.87	0.86	0.78	0.91	0.60
	Corrected	0.64	0.74	0.85	0.80	0.97	0.96	0.87	0.95	0.87
2	Base	-1.62	0.17	0.72	0.82	0.79	-1.32	-0.05	0.2	0.88
	Corrected	0.48	0.78	0.82	0.93	0.94	0.89	0.86	0.90	0.90
3	Base	-0.55	0.47	0.42	0.21	-0.54	0.61	0.70	0.91	0.65
	Corrected	0.59	0.89	0.89	0.83	0.78	0.92	0.84	0.50	0.93
4	Base	-0.48	0.43	-2.10	0.93	0.36	0.37	0.80	0.56	0.80
	Corrected	0.44	0.94	0.78	0.93	0.88	0.83	0.95	0.95	0.88
5	Base	0.08	0.74	0.85	0.80	0.50	0.41	0.75	0.88	0.7
	Corrected	0.50	0.89	0.87	0.92	0.91	0.89	0.94	0.97	0.97
Global median	Base	-0.55	0.43	0.57	0.80	0.50	0.41	0.75	0.88	0.70
	Corrected	0.50	0.89	0.85	0.92	0.91	0.89	0.87	0.95	0.90

Table 6.9: R2 values exhibited by base and corrected model for each considered well (1-9) for five (5) DOEs

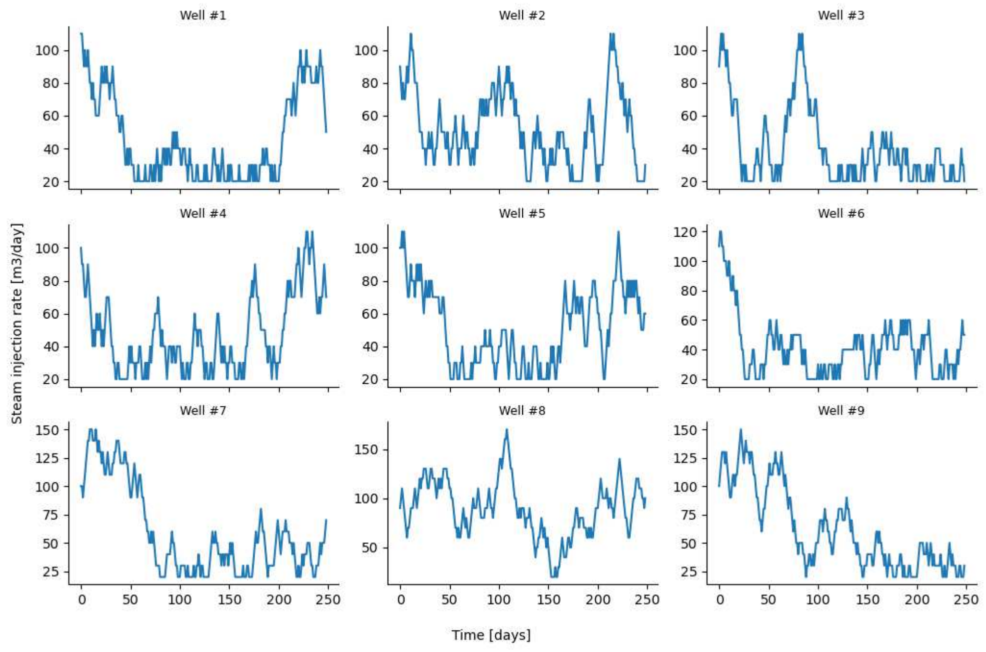


Figure 6.23: Optimal steam injection policy for CS9

6.6 Summary

This chapter proposes a new method for the effective and efficient n-step prediction of time-varying variables of a dynamical system. The method consists of: i) identifying a one-step forecast model of the process that can be used recursively for n-step prediction (*base model*) and ii) modeling the residuals exhibited by the base model. The residual model will act as a correction term of the base model, and the coupling is referred to as the, *corrected model*.

The method was tested on three (3) variants of a multi-well pair numerical reservoir simulation model of SAGD operations, built with publicly available data from heavy oil reservoirs, including information from northern Alberta SAGD operations and CMG-STARS examples. The three (3) variants differed on the number of well pairs considered: one (CS1), three (CS3), and nine (CS9) well pairs.

The methodology extends the conventional surrogate modeling and optimization (SMO) steps: design of experiments, evaluation of each sample using a high-fidelity model, surrogate model identification and validation, and optimization. However, unlike the conventional SMO approach, the objective is to predict a time-varying variable. So, for example, the design of experiments consists in generating a predetermined number of time-varying inputs and the evaluation step involves obtaining the corresponding time-varying outputs.

The methodology was tested on the basis of its capacity to reproduce the true data (numerical reservoir simulation model) - *bias*, and how the sensitivity of the bias to a particular design of experiments - *variance*. The bias was evaluated using R2 and Mean Absolute Percentage Errors (MAPE), and the variance using five (5) independent designs of experiments (DOEs). Results show that,

- Considering a *one well pair*, the corrected model could offer a median R2 of 0.95 and compared to 0.64 for the base model, furthermore the reduction of mean and standard deviation of the distribution of MAPE was 67% and 56%

- For the *three well pair* case, the corrected model was able to show median R2 values of between 0.88 and 0.92 among wells, and a reduction of global MAPE values distribution moments of 49% and 71.2% for the median and standard deviation, respectively.

- Regarding the *nine well pair* case, the corrected model exhibited median R2 values of over 0.85 values for all wells, except for Well 1, which was 0.5. Moreover, a reduction of global MAPE values distribution mean and standard deviation of, 62.9% and 61.6%, respectively.

Chapter 7

Conclusions and Recommendations

7.1 Conclusions

In the context of Steam Assisted Gravity Drainage (SAGD) operations, finding the steam injection rate at every time step or policy, that maximizes long-term cumulative performance represents a major challenge due to complexity of the process. This complexity is manifested in several ways, i) the number of concurrent sub-processes, *e.g.* heat transfer, counter- current flow, co-current flow, ii) potential reservoir heterogeneity, and iii) the lagged nature of the process. In this work, this problem is formulated as an optimal control problem and two (2) machine learning strategies are proposed as possible solutions, *i.e.* **reinforcement learning** and, **dynamic surrogate modeling and optimization**.

Reinforcement learning (RL) In this work, action-value (Chapter 3) and policy gradient (Chapter 4) implementations of RL are presented for the optimization of one well and multi-well SAGD operations, respectively. Furthermore, a qualitative analysis the optimal steam injection polices to gain insight in the SAGD process is given. In both implementations optimal steam injection policies were obtained, exhibiting a significant improvement both with respect to the initial (random) policies and constant steam injection strategies.

For both case studies, optimal steam injection policy exhibit a similar shape; two regions are observed: 1) an increase or slight increase of steam injection rates, and 2) a sharp decrease until reaching the minimum value. A qualitative analysis of these policies suggest that for optimal SAGD operations:

1. Steam chamber expansion is key the overburden is reached (Region 1), afterwards, reservoir temperature should be kept high (Region 2)

2. Pressure plays a vital role until the steam chamber reaches the overburden, afterwards temperature is the driving mechanism of oil production

Dynamic surrogate modeling and optimization (DSMO) Reinforcement learning currently represents one of the most promising methods to solve optimal control problems. However, one of the most important drawbacks when applied to SAGD processes, is that it requires the continuous execution of a potentially computationally expensive numerical reservoir simulation model. To overcome this, we propose a novel method for the effective and efficient prediction of time-varying outputs of a dynamical system that can be used to solve the cited optimization problem.

This method represents an improvement of the conventional recursive based prediction approach. In this approach, a one-step prediction model(s) is identified and then used recursively to predict *n*-steps in the future. The proposed method consists of identifying a second model that can capture the variability of the residual exhibited by the first model, and then act as a correction term. It relies on the assumption the residuals of the recursive approach are correlated with time and thus can be generalized over the input space.

Results suggest that the proposed approach consistently outperforms the conventional recursive approach in terms of *bias* and *variance* (effectiveness). In particular, the proposed or corrected model, exhibits a significant reduction of the errors (*bias*) considering several statistical realizations (*variance*). Moreover, we show that performance is not significantly affected by number of samples, thus proving the efficiency of the method.

More specifically, the major contributions of this research can be summarized as follows:

- Formulate the problem of finding the optimal steam injection policy for SAGD process as an optimal control problem
- Implement and evaluate action value and policy gradient algorithms to solve the cited optimal control problem
- Interpret obtained optimal steam injection policies on the basis of its physical relevance in the context of the SAGD process, *e.g.* role of temperature and pressure
- Develop a new methodology for the construction of dynamic surrogate models for rapid evaluations of SAGD steam injection policies
- Apply the developed methodology to case studies with varying number of well pairs

7.2 Recommendations

Regarding the solution of the SAGD optimization problem, proposed future work includes:

- In a multiwell setting, consider a restriction on the maximum amount of available steam, *i.e.* steam allocation problem. In the RL approach, this can be achieved by giving the agent and artificially low reward when it takes an action that violates the constraints. In DSMO, a the problem becomes a constrained optimization problem that can be solved using conventional optimization methods. This problem has been traditionally solved using real-time SAGD optimization as reported in [92]

- Account for geological uncertainty in a robust optimization approach, *e.g.* maximize mean and minimize standard deviation of the NPV distribution. Although RL by design is able to handle uncertainty as referred in Equation 3.1, due to the computational expense of the reservoir simulation it might not be feasible. However, including geological uncertainty parameters as input for the dynamic surrogate model and then implementing robust optimization is a viable option

- Obtain optimal steam injection policies for different economical parameters (*e.g.* oil price) and provide insight in SAGD optimal operations in for example, high and low oil price scenarios

In terms of the RL implementation possible recommendations are:

- Evaluate the use of other widely used RL algorithms, such as, Actor-Critic Method (A2C, AC3), Deterministic policy gradient, and study results in terms of, objective function optimal value, steam injection policies, number of iterations (or episodes) required, and in deterministic algorithms (*e.g.* policy gradients) the robustness of the learning process

- Increase the number of actions per well could provide the agent more degrees of freedom and could have an important impact on the final optimal policy

Regarding the DSMO approach some suggestions include:

- Extend the work developed in Chapter 5 and consider using a single Multiple Input Multiple Output (MIMO) systems for the one-step forecast models to predict all the required outputs (*e.g.* oil production rates, average pressure) for each well, and perhaps to predict all one-step ahead variables for all the considered wells

- Study the effect of the number of previous steps information on the input of the models

- Extend the framework described in Chapter 6 to predict the full 3D distribution of pressure and saturations over time, as opposed to time series. This could be achieved by adapting the conventional Gradient Boosting framework described in Chapter 1 for dynamic surrogate modeling. In particular, using M weak learners, where each learner will be modeled on the residuals exhibited by the previous one. Additionally, leveraging convolutional neural networks for the 3D distributions for the M weak learners

References

[1] B. Abdulhai, R. Pringle, and G. J. Karakoulas, “Reinforcement learning for true adaptive traffic signal control,” *Journal of Transportation Engineering*, vol. 129, no. 3, pp. 278–285, 2003.

[2] N. Aissani, B. Beldjilali, and D. Trentesaux, “Dynamic scheduling of maintenance tasks in the petroleum industry: A reinforcement approach,” *Engineering Applications of Artificial Intelligence*, vol. 22, no. 7, pp. 1089–1103, 2009.

[3] I. Aizenberg, L. Sheremetov, L. Villa-Vargas, and J. Martinez-Muñoz, “Multilayer neural network with multi-valued neurons in time series forecasting of oil production,” *Neurocomputing*, vol. 175, pp. 980–989, 2016.

[4] W. Ampomah, R. Balch, M. Cather, *et al.*, “Optimum design of CO2 storage and oil recovery under geological uncertainty,” *Applied Energy*, vol. 195, pp. 80–92, 2017.

[5] G. Aydin, “Production modeling in the oil and natural gas industry: An application of trend analysis,” *Petroleum Science and Technology*, vol. 32, no. 5, pp. 555–564, 2014.

[6] D. Baghernezhad, M. Siavashi, and A. Nakhaee, “Optimal scenario design of steam-assisted gravity drainage to enhance oil recovery with temperature and rate control,” *Energy*, vol. 166, pp. 610–623, 2019.

[7] G. Bontempi, S. Ben Taieb, and Y.-A. L. Borgne, “Machine learning strategies for time series forecasting,” in *European business intelligence summer school*, Springer, 2012, pp. 62–77.

[8] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[9] R. Butler, “A new approach to the modelling of steam-assisted gravity drainage,” *Journal of Canadian Petroleum Technology*, vol. 24, no. 03, pp. 42–51, 1985.

[10] R. M. Butler, *Thermal recovery of oil and bitumen.* Prentice Hall Englewood Cliffs, NJ, 1991, vol. 46.

[11] E. Carrero, N. V. Queipo, S. Pintos, and L. E. Zerpa, "Global sensitivity analysis of alkali–surfactant–polymer enhanced oil recovery processes," *Journal of Petroleum Science and Engineering*, vol. 58, no. 1-2, pp. 30–42, 2007.

[12] "Cenovus Christina Lake In-Situ Oil Scheme 2010 -2011 Update," Cenovus Energy, Tech. Rep., 2010.

[13] B. Chen and A. C. Reynolds, "CO2 water-alternating-gas injection for enhanced oil recovery: Optimal well controls and half-cycle lengths," *Computers & Chemical Engineering*, vol. 113, pp. 44–56, 2018.

[14] G. Chen, K. Zhang, X. Xue, *et al.*, "Surrogate-assisted evolutionary algorithm with dimensionality reduction method for water flooding production optimization," *Journal of Petroleum Science and Engineering*, vol. 185, p. 106 633, 2020.

[15] G. Chen, K. Zhang, L. Zhang, *et al.*, "Global and local surrogate-model-assisted differential evolution for waterflooding production optimization," *SPE Journal*, vol. 25, no. 01, pp. 105–118, 2020.

[16] T. Chouard. "The Go files: AI computer clinches victory against Go champion." (2016), [Online]. Available: `https://www.nature.com/news/the-go-files-ai-computer-clinches-victory-against-go-champion-1.19553`. (accessed: 04.17.2020).

[17] K. Chung and R. Butler, "Geometrical effect of steam injection on the formation of emulsions nn the steam-assisted gravity drainage process," *Journal of Canadian Petroleum Technology*, vol. 27, no. 01, 1988.

[18] Computer Modeling Group, *CMOST Users Manual; Version 2018.10.* CMG Ltd Calgary, Canada, 2018.

[19] Computer Modeling Group, *STARS Users Manual; Version 2018.10.* CMG Ltd Calgary, Canada, 2018.

[20] K. Dalamagkidis, D. Kolokotsa, K. Kalaitzakis, and G. S. Stavrakakis, "Reinforcement learning for energy conservation and comfort in buildings," *Building and environment*, vol. 42, no. 7, pp. 2686–2698, 2007.

[21] G. De Paola, C. Ibanez-Llano, J. Rios, and G. Kollias, "Reinforcement learning for field development policy optimization," in *SPE Annual Technical Conference and Exhibition*, Society of Petroleum Engineers, 2020.

[22] N. Edmunds, "On the difficult birth of SAGD," *Journal of Canadian Petroleum Technology*, vol. 38, no. 01, 1999.

[23] N. Edmunds, B. Moini, and J. Peterson, "Advanced solvent-additive processes by genetic optimization," *Journal of Canadian Petroleum Technology*, vol. 49, no. 09, pp. 34–41, 2010.

[24] N. R. Edmunds, "Investigation of sagd steam trap control in two and three dimensions," in *SPE international conference on horizontal well technology*, OnePetro, 1998.

[25] G. van Essen, P. Van den Hof, and J.-D. Jansen, "A two-level strategy to realize life-cycle production optimization in an operational setting," *SPE Journal*, vol. 18, no. 06, pp. 1–057, 2013.

[26] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[27] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: A statistical view of boosting (with discussion and a rejoinder by the authors)," *The annals of statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[28] E. Gallardo and C. V. Deutsch, "Approximate physics-discrete simulation of the steam-chamber evolution in steam-assisted gravity drainage," *SPE Journal*, vol. 24, no. 02, pp. 477–491, 2019.

[29] Y. Gao, M. Chen, B. Lin, and Y. Jin, "Modeling of reservoir temperature upon preheating in sagd wells considering phase change of bitumen," *International Journal of Heat and Mass Transfer*, vol. 144, p. 118 650, 2019.

[30] I. D. Gates, J. Kenny, I. L. Hernandez-Hdez, and G. L. Bunio, "Steam injection strategy and energetics of steam-assisted gravity drainage," in *SPE International Thermal Operations and Heavy Oil Symposium*, Society of Petroleum Engineers, 2005.

[31] I. Gates and N. Chakrabarty, "Optimization of steam assisted gravity drainage in mcmurray reservoir," *Journal of Canadian Petroleum Technology*, vol. 45, no. 09, 2006.

[32] A. Golzari, M. H. Sefat, and S. Jamshidi, "Development of an adaptive surrogate model for production optimization," *Journal of petroleum Science and Engineering*, vol. 133, pp. 677–688, 2015.

[33] G. J. Gordon, "Reinforcement learning with function approximation converges to a region," in *Advances in neural information processing systems*, 2001, pp. 1040–1046.

[34] G. J. Gordon, "Chattering in sarsa(lambda) - a cmu learning lab internal report," Tech. Rep., 1996.

[35] D. R. Gotawala and D. Gates, "Sagd subcool control with smart injection wells," in *EUROPEC/EAGE Conference and Exhibition*, OnePetro, 2009.

[36] D. R. Gotawala and I. D. Gates, "Steam fingering at the edge of a steam chamber in a heavy oil reservoir," *The Canadian Journal of Chemical Engineering*, vol. 86, no. 6, pp. 1011–1022, 2008.

[37] P. A. Govind, S. Das, S. Srinivasan, and T. Wheeler, "Expanding solvent sagd in heavy oil reservoirs," in *Society of Petroleum Engineers-International Thermal Operations and Heavy Oil Symposium, ITOHOS 2008-" Heavy Oil: Integrating the Pieces"*, Society of Petroleum Engineers, 2008, pp. 541–554.

[38] J. Guevara, R. Patel, and J. Trivedi, "Optimization of steam injection in SAGD using reinforcement learning," *Journal of Petroleum Science and Engineering*, vol. 206, p. 108 735, 2021.

[39] J. Guevara, R. G. Patel, and J. J. Trivedi, "Optimization of steam injection for heavy oil reservoirs using reinforcement learning," in *SPE International Heavy Oil Conference and Exhibition*, Society of Petroleum Engineers, 2018.

[40] J. Guevara and J. Trivedi, "Towards a machine learning based dynamic surrogate modeling and optimization of steam injection policy in SAGD," in *SPE Western Regional Meeting*, OnePetro, 2022.

[41] T. Guo, J. Wang, and I. D. Gates, "Pad-scale control improves SAGD performance," *Petroleum*, vol. 4, no. 3, pp. 318–328, 2018.

[42] R. Guseo, C. Mortarino, and M. A. Darda, "Homogeneous and heterogeneous diffusion models: Algerian natural gas production," *Technological Forecasting and Social Change*, vol. 90, pp. 366–378, 2015.

[43] L. J. Herrera, H. Pomares, I. Rojas, A. Guillén, A. Prieto, and O. Valenzuela, "Recursive prediction for long term time series forecasting using advanced models," *Neurocomputing*, vol. 70, no. 16-18, pp. 2870–2880, 2007.

[44] B. Horowitz, S. M. B. Afonso, and C. V. P. de Mendonça, "Surrogate based optimal waterflooding management," *Journal of Petroleum Science and Engineering*, vol. 112, pp. 206–219, 2013.

[45] F. Hourfar, H. J. Bidgoly, B. Moshiri, K. Salahshoor, and A. Elkamel, "A reinforcement learning approach for waterflooding optimization in petroleum reservoirs," *Engineering Applications of Artificial Intelligence*, vol. 77, pp. 98–116, 2019.

[46] F. Hourfar, B. Moshiri, K. Salahshoor, M. Zaare-Mehrjerdi, and P. Pourafshary, "Adaptive modeling of waterflooding process in oil reservoirs," *Journal of petroleum Science and Engineering*, vol. 146, pp. 702–713, 2016.

[47] H. R. Jahangiri and D. Zhang, "Ensemble based co-optimization of carbon dioxide sequestration and enhanced oil recovery," *International Journal of Greenhouse Gas Control*, vol. 8, pp. 22–33, 2012.

[48] J. Jansen, "Adjoint-based optimization of multi-phase flow through porous media–a review," *Computers & Fluids*, vol. 46, no. 1, pp. 40–51, 2011.

[49] P. Jiang, Q. Zhou, and X. Shao, "Surrogate-model-based design and optimization," in *Surrogate Model-Based Engineering Design and Optimization*, Springer, 2020, pp. 135–236.

[50] S. Joe and F. Y. Kuo, "Constructing sobol sequences with better two-dimensional projections," *SIAM Journal on Scientific Computing*, vol. 30, no. 5, pp. 2635–2654, 2008.

[51] A. Kumar and H. Hassanzadeh, "A qualitative study of the impact of random shale barriers on SAGD performance using data analytics and machine learning," *Journal of Petroleum Science and Engineering*, vol. 205, p. 108 950, 2021.

[52] A. Kumar and H. Hassanzadeh, "Impact of shale barriers on performance of SAGD and ES-SAGD—a review," *Fuel*, vol. 289, p. 119 850, 2021.

[53] B. Liang, J. Liu, J. You, J. Jia, Y. Pan, and H. Jeong, "Hydrocarbon production dynamics forecasting using machine learning: A state-of-the-art review," *Fuel*, vol. 337, p. 127 067, 2023.

[54] B. Lim and S. Zohren, "Time-series forecasting with deep learning: A survey," *Philosophical Transactions of the Royal Society A*, vol. 379, no. 2194, p. 20 200 209, 2021.

[55] H. Liu, D. Zhu, Y. Liu, A. Du, D. Chen, and Z. Ye, "A reinforcement learning based 3D guided drilling method: Beyond ground control," in *Proceedings of the 2018 VII International Conference on Network, Communication and Computing*, 2018, pp. 44–48.

[56] L. Ljung, "System identification," in *Signal analysis and prediction*, Springer, 1998, pp. 163–173.

[57] D. G. Luenberger, Y. Ye, *et al.*, *Linear and nonlinear programming.* Springer, 1984, vol. 2.

[58] H. Ma, G. Yu, Y. She, and Y. Gu, "Waterflooding optimization under geological uncertainties by using deep reinforcement learning algorithms," in *SPE Annual Technical Conference and Exhibition*, Society of Petroleum Engineers, 2019.

[59] S. Makridakis and M. Hibon, "Arma models and the box–jenkins methodology," *Journal of forecasting*, vol. 16, no. 3, pp. 147–163, 1997.

[60] Martín Abadi, Ashish Agarwal, Paul Barham, *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: `https://www.tensorflow.org/`.

[61] K. McBride and K. Sundmacher, "Overview of surrogate modeling in chemical process engineering," *Chemie Ingenieur Technik*, vol. 91, no. 3, pp. 228–239, 2019.

[62] M. D. McKay, R. J. Beckman, and W. J. Conover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 42, no. 1, pp. 55–61, 2000.

[63] P. Meum, P. Tøndel, J.-M. Godhavn, and O. M. Aamo, "Optimization of smart well production through nonlinear model predictive control," in *Intelligent Energy Conference and Exhibition*, OnePetro, 2008.

[64] R. Miftakhov, A. Al-Qasim, and I. Efremov, "Deep reinforcement learning: Reservoir optimization from pixels," in *International Petroleum Technology Conference*, International Petroleum Technology Conference, 2020.

[65] P. Mishra, "Forecasting natural gas price-time series and nonparametric approach," in *Proceedings of the World Congress on Engineering*, WCE, vol. 1, 2012, pp. 4–6.

[66] Y. Mohankumar, Z. Li, and B. Huang, "Steam allocation and production optimization in SAGD reservoir under steam-to-oil ratio uncertainty," *Journal of Petroleum Science and Engineering*, vol. 183, p. 106 456, 2019.

[67] A. Moolya, A. Rodrıguez-Martınez, and I. E. Grossmann, "Optimal producer well placement and multiperiod production scheduling using surrogate modeling," *Computers & Chemical Engineering*, vol. 165, p. 107 941, 2022.

[68] T. M. Moussa and A. A. Awotunde, "Self-adaptive differential evolution with a novel adaptation technique and its application to optimize es-sagd recovery process," *Computers & Chemical Engineering*, vol. 118, pp. 64–76, 2018.

[69] T. Nasr, D. Law, H. Golbeck, and G. Korpany, "Counter-current aspect of the SAGD process," *Journal of Canadian Petroleum Technology*, vol. 39, no. 01, 2000.

[70] B. M. Negash, L. D. Tufa, M. Ramasamy, and M. B. Awang, "System identification based proxy model of a reservoir under water injection," *Modelling and Simulation in Engineering*, vol. 2017, 2017.

[71] A. Y. Ng, A. Coates, M. Diel, *et al.*, "Autonomous inverted helicopter flight via reinforcement learning," in *Experimental robotics IX*, Springer, 2006, pp. 363–372.

[72] C. S. W. Ng, A. J. Ghahfarokhi, and M. N. Amar, "Production optimization under waterflooding with long short-term memory and metaheuristic algorithm," *Petroleum*, 2022.

[73] P. Ogbeiwi, K. D. Stephen, and A. O. Arinkoola, "Robust optimisation of water flooding using an experimental design-based surrogate model: A case study of a niger-delta oil reservoir," *Journal of Petroleum Science and Engineering*, vol. 195, p. 107 824, 2020.

[74] C. Olah. "Understanding lstm networks." (2015), [Online]. Available: `https://colah.github.io/posts/2015-08-Understanding-LSTMs/` (visited on 09/14/2022).

[75] R. G. Patel, V. Prasad, and J. J. Trivedi, "Real-time production optimization of steam-assisted-gravity-drainage reservoirs using adaptive and gain-scheduled model-predictive control: An application to a field model," *SPE Production & Operations*, vol. 34, no. 01, pp. 72–89, 2019.

[76] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[77] N. V. Queipo, J. V. Goicochea, and S. Pintos, "Surrogate modeling-based optimization of sagd processes," *Journal of Petroleum Science and Engineering*, vol. 35, no. 1-2, pp. 83–93, 2002.

[78] N. V. Queipo, R. T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P. K. Tucker, "Surrogate-based analysis and optimization," *Progress in aerospace sciences*, vol. 41, no. 1, pp. 1–28, 2005.

[79] S. Razavi, B. A. Tolson, and D. H. Burn, "Review of surrogate modeling in water resources," *Water Resources Research*, vol. 48, no. 7, 2012.

[80] G. Reddy, A. Celani, T. J. Sejnowski, and M. Vergassola, "Learning to soar in turbulent environments," *Proceedings of the National Academy of Sciences*, vol. 113, no. 33, E4877–E4884, 2016.

[81] P. E. Rose, *The steam-assisted gravity drainage of oil sand bitumen*. The University of Utah, 1993.

[82] B. Rubin and W. L. Buchanan, "A general purpose thermal model," *Society of Petroleum Engineers Journal*, vol. 25, no. 02, pp. 202–214, 1985.

[83] Ryan Solgi, *Geneticalgorithm*, 2022. [Online]. Available: `https://pypi.org/project/geneticalgorithm/`.

[84] M. Salehian, M. H. Sefat, and K. Muradov, "Multi-solution well placement optimization using ensemble learning of surrogate models," *Journal of Petroleum Science and Engineering*, vol. 210, p. 110 076, 2022.

[85] L. Saputelli, M. Nikolaou, and M. Economides, "Real-time reservoir management: A multiscale adaptive optimization and control approach," *Computational Geosciences*, vol. 10, no. 1, pp. 61–96, 2006.

[86] Schlumberger, *Eclipse Reference Manual; Version 2014.10*. 2014.

[87] L. B. Sheremetov, A. González-Sánchez, I. López-Yáñez, and A. V. Ponomarev, "Time series forecasting: Applications to the upstream oil and gas supply chain," *IFAC Proceedings Volumes*, vol. 46, no. 9, pp. 957–962, 2013.

[88] J. Shi and J. Y. Leung, "Physics-based proxy modelling of solvent transport in vapex process," *The Canadian Journal of Chemical Engineering*, vol. 92, no. 8, pp. 1467–1480, 2014.

[89] J. Shi, V. Vishal, and J. Y. Leung, "Uncertainty assessment of vapex performance in heterogeneous reservoirs using a semi-analytical proxy model," *Journal of Petroleum Science and Engineering*, vol. 122, pp. 290–303, 2014.

[90] H. Shijun, X. Hao, W. Shaolei, H. Chenghui, and Y. Yang, "Physical simulation of the interlayer effect on SAGD production in mackay river oil sands," *Fuel*, vol. 183, pp. 373–385, 2016.

[91] A. Shokry, P. Baraldi, E. Zio, and A. Espuña, "Dynamic surrogate modeling for multistep-ahead prediction of multivariate nonlinear chemical processes," *Industrial & Engineering Chemistry Research*, vol. 59, no. 35, pp. 15 634–15 655, 2020.

[92] N. Sibaweihi, "Risk management and optimization in real-time noncondensable gas co-injection under economic uncertainty," *SPE Reservoir Evaluation & Engineering*, pp. 1–21, 2022.

[93] P. Siddhamshetty, P. Bhandakkar, and J. S.-I. Kwon, "Enhancing total fracture surface area in naturally fractured unconventional reservoirs via model predictive control," *Journal of Petroleum Science and Engineering*, vol. 184, p. 106 525, 2020.

[94] D. Silver, J. Schrittwieser, K. Simonyan, *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[95] A. Sorjamaa, J. Hao, N. Reyhani, Y. Ji, and A. Lendasse, "Methodology for long-term prediction of time series," *Neurocomputing*, vol. 70, no. 16-18, pp. 2861–2869, 2007.

[96] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[97] A. Y. Sun, "Optimal carbon storage reservoir management through deep reinforcement learning," *Applied Energy*, vol. 278, p. 115 660, 2020.

[98] Z. Sun, J. Xu, D. N. Espinoza, and M. T. Balhoff, "Optimization of subsurface CO2 injection based on neural network surrogate modeling," *Computational Geosciences*, vol. 25, no. 6, pp. 1887–1898, 2021.

[99] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction.* MIT press, 2018.

[100] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in neural information processing systems*, 2000, pp. 1057–1063.

[101] S. B. Taieb and G. Bontempi, "Recursive multi-step time series forecasting by perturbing data," in *2011 IEEE 11th International Conference on Data Mining*, IEEE, 2011, pp. 695–704.

[102] E.-G. Talbi, *Metaheuristics: from design to implementation.* John Wiley & Sons, 2009.

[103] J. F. Torres, D. Hadjout, A. Sebaa, F. Martınez-Álvarez, and A. Troncoso, "Deep learning for time series forecasting: A survey," *Big Data*, vol. 9, no. 1, pp. 3–21, 2021.

[104] J. Van Doren, S. Douma, B. Wassing, H. Kraaijevanger, and B.-R. de Zwart, "Adjoint-based optimization of polymer flooding," in *SPE Enhanced Oil Recovery Conference*, OnePetro, 2011.

[105] G. Van Essen, M. Zandvliet, P. Van den Hof, O. Bosgra, and J.-D. Jansen, "Robust waterflooding optimization of multiple geological scenarios," *SPE Journal*, vol. 14, no. 01, pp. 202–210, 2009.

[106] J. W. Vanegas, L. Cunha, and C. V. Deutsch, "Proxy models for fast transfer of static uncertainty to reservoir performance uncertainty," 2011.

[107] S. S. Vembadi, R. G. Patel, J. J. Trivedi, and V. Prasad, "Real-time feedback control of sagd wells using model predictive control to optimize steam chamber development under uncertainty," *The Canadian Journal of Chemical Engineering*, vol. 96, no. 6, pp. 1290–1305, 2018.

[108] A. Venkatraman, M. Hebert, and J. A. Bagnell, "Improving multi-step prediction of learned time series models," in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[109] T.-T. Wong, W.-S. Luk, and P.-A. Heng, "Sampling with hammersley and halton points," *Journal of graphics tools*, vol. 2, no. 2, pp. 9–24, 1997.

[110] C. Yang and X. Wang, "A steam injection distribution optimization method for SAGD oil field using lstm and dynamic programming," *ISA transactions*, vol. 110, pp. 198–212, 2021.

[111] S. Yang, Z. Nie, S. Wu, *et al.*, "A critical review of reservoir simulation applications in key thermal recovery processes: Lessons, opportunities, and challenges," *Energy & Fuels*, vol. 35, no. 9, pp. 7387–7405, 2021.

[112] S. Yao, J. J. Trivedi, and V. Prasad, "Proxy modeling of the production profiles of sagd reservoirs based on system identification," *Industrial & Engineering Chemistry Research*, vol. 54, no. 33, pp. 8356–8367, 2015.

[113] J. Yu and B. Jafarpour, "Active learning for well control optimization with surrogate models," *SPE Journal*, pp. 1–21, 2022.

[114] J.-Y. Yuan and D. Nugent, "Subcool, fluid productivity, and liquid level above a sagd producer," *Journal of Canadian Petroleum Technology*, vol. 52, no. 05, pp. 360–367, 2013.

[115] H. Zanbouri and K. Salahshoor, "Development of robust surrogate model for economic performance prediction of oil reservoir production under waterflooding process," *Journal of Petroleum Science and Engineering*, vol. 165, pp. 496–504, 2018.

[116] M. Zandvliet, M. Handels, G. Van Essen, D. Brouwer, and J.-D. Jansen, "Adjoint-based well-placement optimization under production constraints," *SPE Journal*, vol. 13, no. 04, pp. 392–399, 2008.

[117] K. Zhang, Z. Wang, G. Chen, *et al.*, "Training effective deep reinforcement learning agents for real-time life-cycle production optimization," *Journal of Petroleum Science and Engineering*, vol. 208, p. 109 766, 2022.