Efficient Optimization and Robust Value Quantification of Enhanced Oil Recovery Strategies

by

Micheal Babatunde Oguntola

Thesis submitted in fulfilment of
the requirements for the degree of
PHILOSOPHIAE DOCTOR
(PhD)

Universitetet
i Stavanger

Faculty of Science and Technology
Department of Energy Resources
2021

University of Stavanger
NO-4036 Stavanger
NORWAY

www.uis.no

ISBN: 978-82-8439-088-8
ISSN: 1890-1387
PhD: Thesis UiS No. 649

Acknowledgment

Words are not enough to express my profound gratitude to the Almighty God for giving me the strength and grace to successfully undergo this PhD training. To Him alone be all the glory. Moreover, the accomplishment of this PhD work would have been impossible without the kind support, constant encouragement, and contribution of some people.

First, I would like to thank my supervisors, Dr. Rolf Lorentzen at the NORCE-Norwegian Research Centre, Dr. Robert Klöfkorn at Lund University, and Prof Aksel Hiorth at the University of Stavanger for the opportunity to undergo this PhD training and also for their constant encouragement. A special thanks to Dr. Rolf Lorentzen for his exceptional supervision, kind support, and valuable feedback throughout this PhD work. I am also grateful to Dr. Robert Klöfkorn for his sincere help, valuable discussions, and research connection at the University of Münster, Germany.

My sincere appreciation goes to Dr. Randi Valestrand, the manager of the Data Assimilation and Optimization group at NORCE, for her efficient assistance and leadership. Also, I would like to thank my colleagues in the energy department at NORCE for their constant motivation, support, and time spent together.

I would like to acknowledge funding from the Research Council of Norway and the industry partners, ConocoPhillips Skandinavia AS, Aker BP ASA, Vår Energi AS, Equinor Energy AS, Neptune Energy Norge AS, Lundin Energy Norway AS, Halliburton AS, Schlumberger Norge AS, and Wintershall Dea Norge AS, of The National IOR Centre of Norway.

I sincerely appreciate prof Mario Ohlberger for the opportunity to undergo a three-month research stay at the Cluster of Excellence Mathematics, University of Münster. To my colleagues, Tim Keil and Hendrik Kleikamp at the University of Münster, your kind support and collaboration are appreciated.

Finally, I dearly acknowledge the effort of my parents and siblings for their constant prayers, communication, and emotional support. My sincere gratitude goes to my partner Tanteliniaina Mioratina for her constant prayers, motivation, and enthusiastic assistance.

Micheal Babatunde Oguntola
Stavanger, March 2022

Abstract

With an increasing demand for hydrocarbon reservoir produces such as oil, etc., and difficulties in finding green oil fields, the use of Enhanced Oil Recovery (EOR) methods such as polymer, Smart water, and solvent flooding for further development of existing fields can not be overemphasized. For reservoir profitability and reduced environmental impact, it is crucial to consider appropriate well control settings of EOR methods for given reservoir characterization. Moreover, finding appropriate well settings requires solving a constrained optimization problem with suitable numerical solution methods. Conventionally, the solution method requires many iterations involving several computationally demanding function evaluations before convergence to the appropriate near optimum. The major subject of this thesis is to develop an efficient and accurate solution method for constrained optimization problems associated with EOR methods for their value quantifications and ranking in the face of reservoir uncertainties.

The first contribution of the thesis develops a solution method based on the inexact line search method (with Ensemble Based Optimization (EnOpt) for approximate gradient computation) for robust constrained optimization problems associated with polymer, Smart water, and solvent flooding. Here, the objective function is the expectation of the Net Present Value (NPV) function over given geological realizations. For a given set of well settings, the NPV function is defined based on the EOR simulation model, which follows from an appropriate extension of the black-oil model. The developed solution method is used to find the economic benefits and also the ranking of EOR methods for different oil reservoirs developed to mimic North Sea reservoirs.

Performing the entire optimization routine in a transformed domain along with truncations has been a common practice for handling simple linear constraints in reservoir optimization. Aside from the fact that this method has a negative impact on the quality of gradient computation, it is complicated to use for non-linear constraints. The second contribution of this thesis proposes a technique based on the exterior penalty method for handling general linear and non-linear constraints in reservoir optimization problems to improve gradient computation quality by the EnOpt method for efficient and improved optimization algorithm.

Because of the computationally expensive NPV function due to the costly reservoir simulation of EOR methods, the solution method for the underlying EOR optimization problem becomes inefficient, especially for large reservoir problems. To speedup the overall computation of the solution method, this thesis introduces a novel full order model (FOM)-based certified adaptive machine learning optimization procedures to locally approximate the expensive NPV function. A supervised feedforward deep neural network (DNN) algorithm is employed to locally create surrogate model. In the FOM-based optimization algorithm of this study, several FOM NPV function evaluations are required by the EnOpt method to approximate the gradient function at each (outer) iteration until convergence. To limit the number FOM-based evaluations, we consider building surrogate models locally to replace the FOM based NPV function at each outer iteration and proceed with an inner optimization routine until convergence. We adapt the surrogate model using some FOM-based criterion where necessary until convergence. The demonstration of methodology for polymer optimization problem on a benchmark model results in an improved optimum and found to be more efficient compared to using the full order model optimization procedures.

List of papers

Paper I

Micheal B. Oguntola and Rolf J. Lorentzen (2020). On the Robust Value Quantification of Polymer EOR Injection Strategies for Better Decision Making. *In ECMOR XVII (Vol. 2020, No.1, pp. 1-25).* European Association of Geoscientists & Engineers. https://doi.org/10.3997/2214-4609.202035057

Paper II

Micheal B. Oguntola and Rolf J. Lorentzen (2021). Ensemble-based constrained optimization using an exterior penalty method. *Journal of Petroleum Science and Engineering,* 207, 109165.

https://doi.org/10.1016/j.petrol.2021.109165

Paper III

Micheal B. Oguntola and Rolf J. Lorentzen (2021). Robust Value Quantification of Enhanced Oil Recovery Methods using Ensemble-Based Optimization. *SPE Journal of Reservoir Engineering.* https://doi.org/10.2118/209587-PA

Paper IV

Timi Godfreg, Hendrik Kleikamp, Rolf J. Lorentzen, Micheal B. Oguntola, Mario Ohlberger (2022). Adaptive machine learning based surrogate modeling to accelerate PDE-constrained optimization for enhanced oil recovery. *Journal of Advances in Computational Mathematics* [under review].

https://doi.org/10.48550/arXiv.2203.01674.

Table of contents

List of figures

Nomenclature

Symbol	Description	Unit
A_α	Accumulation term for fluid phase α	-
b_α	Inverse of volume formation factor for phase α	$(\mathrm{m}^3/\mathrm{stdm}^3)^{-1}$
c	Polymer concentration	$\mathrm{kg/m}^3$
c^a	Polymer adsorption concentration	$\mathrm{kg/m}^3$
$P_{c,gw}$	Capillary pressure at gas-water interface	Bar
$P_{c,og}$	Capillary pressure at oil-gas interface	Bar
$P_{c,ow}$	Capillary pressure at oil-water interface	Bar
$\partial\Omega$	Boundary of reservoir domain	–
$\mathbf{a}$	Acceleration due to gravity	$\mathrm{m/d}^2$
J	Reservoir net present value function	USD
$\mathbf{K}$	Permeability tensor	m^2
c^*	Maximum allowable polymer concentration	$\mathrm{kg/m}^3$
$\mathbb{N}$	Set of natural numbers	–
Ω	Reservoir domain	–
ω	Todd-Longstaff mixing parameter	–
ϕ	Rock porosity	-
p_α	Pressure of phase α	Bar

Q_α	Well outflux density of fluid phase (or component) α based on the well model	kg/(m^{3}d)
r_{og}	Ratio of dissolved oil to gas in the gaseous phase	m^3/stdm^3
$\mathbb{R}$	Set of real numbers	–
k_{rs}	Relative permeability of solvent	–
ρ_α	Density of phase or component α	kg/m^3
ρ_r	Reservoir rock density	kg/m^3
ρ_s	Density of solvent	kg/m^3
R_k	Permeability reduction quantification due polymer flooding	–
r_{go}	Ratio of dissolved gas to oil in the oleic phase	stdm^3/stdm^3
S_α	Saturation of phase α	-
c_s	Salinity of Smart water	kg/m^3
S_s	Saturation of solvent	–
S_{ipv}	Fraction of the reservoir pore volume that is inaccessible	–
t	Time	d
$\boldsymbol{\tau}_\alpha$	Surface volume flux of fluid phase α (oriented quantity)	-
$\mathbf{v}_\alpha$	Darcy flux of phase (or component) α	m/d
$\mu_{s,\text{eff}}$	Effective viscosity of salt	cp
$\mu_{p,\text{eff}}$	Effective viscosity of polymer solution	cp

$\mu_{\text{fmp}}(c^*)$	Viscosity of fully mixed polymer solution containing maximum allowable polymer concentration c^*	cp
μ_{fmp}	Viscosity of partially mixed water with polymer	cp
$k_{r\alpha}$	Relative permeability of phase α	–
μ_α	viscosity of phase (or component) α	cp
$\mathbf{X}$	Reservoir dynamic state vector	-

List of abbreviations

CMA	Covariance Matrix Adaptation
EnOpt	Ensemble Based Optimization
EOR	Enhanced Oil Recovery
EPF	Exterior Penalty Function
FDSA	Finite Difference Stochastic Approximation
MCGA	Monte Carlo Gradient Approximation
MMP	Minimum Miscible Pressure
NCS	Norwegian Continental Shelf
NPV	Net Present Value
OOIP	Original Oil in Place
OPM	Open Porus Media
ORF	Oil Recovery Factor
ReLU	Rectified Linear Unit
REORO	Robust EOR Optimization
ROM	Reduced Order Model
SPSA	Simultaneous Perturbation Stochastic Approximatio
StoSAG	Stochastic Simplex Approximate Gradient

Part I

Scientific background

1 Introduction

1.1 Petroleum production optimization

To meet the continuous increase in the world energy demand resulting from the rise in world population and the quest for a high standard of living, the need to utilize traditional energy resources such as petroleum (or simply oil) and natural gas will continue. The dependence becomes more pronounced, especially in developing countries where infrastructures for alternative sources of energy such as solar, wind, etc., are not easily accessible due to high cost and implementation resources in the near future. Petroleum is a finite energy resource embedded in an interconnected network of porous rock formations called the petroleum reservoir.

In practice, the oil is recovered from reservoirs by different methods based on reservoir complexities, e.g., the conventional water flooding method, where water is injected into the reservoir to improve driving force, see, Figure 1.1. However, with the conventional oil recovery method, studies have shown that the volume of residual oil is still very high after production cessation. For instance, see Figure 1.2, which summarizes the results of studies conducted on 29 large oil reservoirs on the Norwegian Continental Shelf (NCS) indicating a sizable percentage of the Original Oil in Place (OOIP) as leftover. Practically, the high leftover can be due to several factors such as limited technical capability, oil recovery method incompatibility with reservoir type, and/ or ineffective production process. Furthermore, as the number of newly discovered green reservoir fields declines while the energy demand increases, many advanced recovery technologies such as the EOR methods are developed to improve the performance of conventional oil recovery methods.

Many research including Fonseca et al. (2017); Jansen (2011b); Kraaijevanger et al. (2007); Wang et al. (2010); Li-xin and Jian-jun (2005), and Liu and Reynolds (2014) extensively studied the actual reservoir profitability (in terms of NPV, Oil Recovery Factor (ORF), etc.) with the conventional oil recovery method using numerical optimization and time-consuming reservoir model simulation. However with the EOR methods, little work has been done in finding the actual economic benefit for field applications under reservoir uncertainties. In this case, the selection of an optimally and appropriate EOR strategy based on

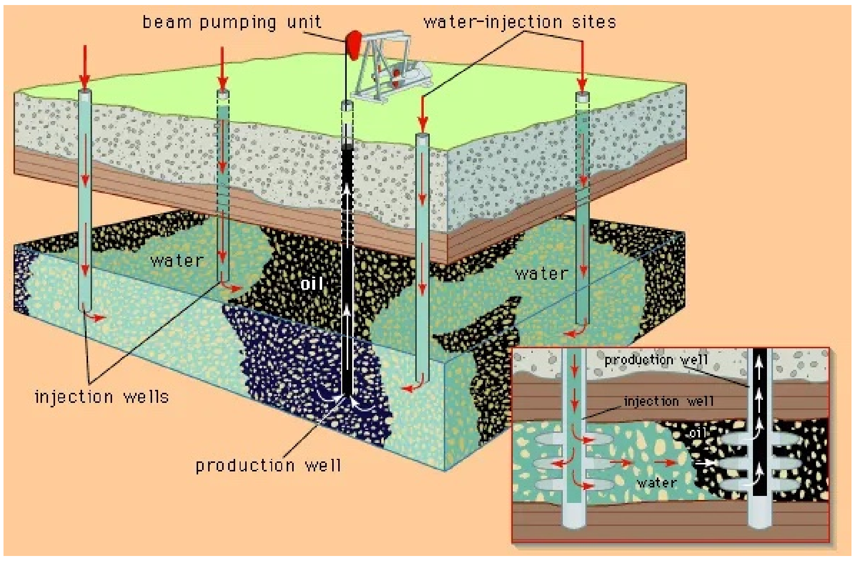


Figure 1.1: Schematic diagram of a quarter of five oil reservoir development using water flooding (Baker, 1998).

reservoir characteristics has not been fully examined. In a petroleum development plan, the problem of reservoir profitability is usually formulated as a constrained optimization (petroleum production optimization) problem. The solution methods coupled with the inefficient reservoir simulation often used for this class of constrained problem give rise to inefficient and less accurate economic prediction techniques. Therefore, solutions to these prevailing problems are researched in this Ph.D. project.

1.2 Motivation

Water flooding remains the most economically utilized secondary oil recovery method after the primary depletion of the reservoir (Ogbeiwi et al., 2018). However, the method is faced with many setbacks, specifically when utilized in medium to highly heterogeneous or unfavorable mobility reservoirs. One such setback is early water breakthrough which leaves behind about 65% of unswept oil in the rock formation (Lyons, 2009; Svein and Kleppe, 1992). It is because of different hindrances (based on the petrophysical properties of the oil reservoir) to residual oil flow. Some of which include the high viscosity of residual oil causing unfavorable mobility ratio, high level of heterogeneity (in porosity and permeability),

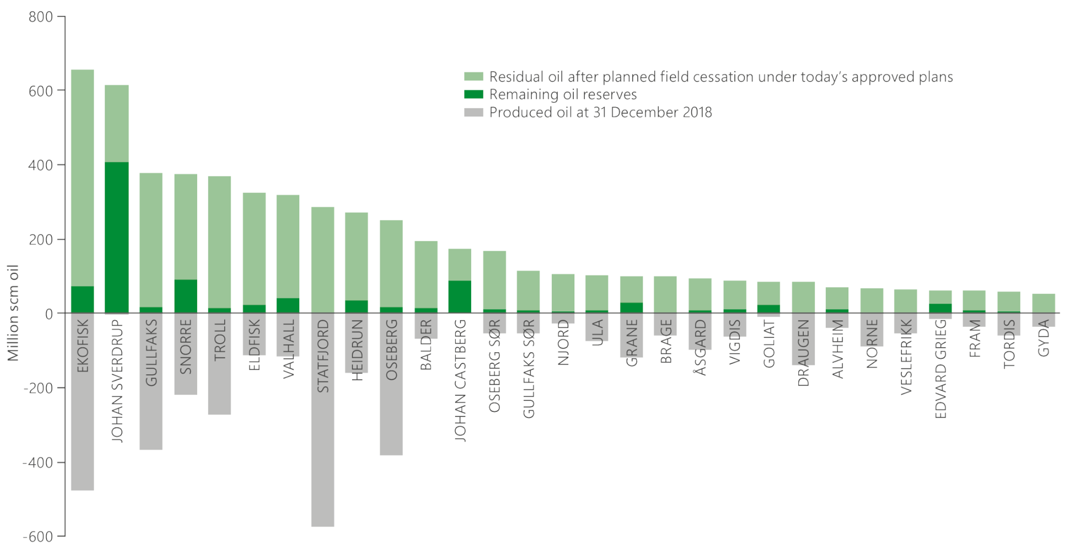


Figure 1.2: Oil produced and residual profiles of 29 reservoir fields on the NCS (Norwegian-Petroleum-Directorate, 2019).

and high interfacial tension between reservoir fluids causing the high capillary forces holding down the oil in the capillary pores of the reservoir (Niu et al., 2020).

The EOR methods are developed to mitigate the effects of hindering factors associated with fluid and rock properties on oil flow in the reservoir by improving the performance of water flooding. The methods involve injecting EOR gas (like CO_2, etc.) or chemical (like polymer, Smart water, etc.) into the reservoir, causing favorable changes to mobility ratio of oil-water system (Xiangguo et al., 2021), rock wettability (Fani et al., 2018), microscopic sweep efficiency (Sehbi et al., 2001), etc., in other to produce more oil. But, because EOR methods are associated with high chemical or gas cost, and also injecting more than necessary into the field can lead to an insignificant increase in oil production, it is imperative to optimize their control strategies for field applications.

Several techniques have been used to find the optimal and low-cost injection strategies for these EOR methods (mostly at laboratory scale) and utilized with different reservoir fields. In this case, the numerical experiment performed mostly involves a single geological reservoir model (Dudek et al., 2021; Xu et al., 2018; Sadeed et al., 2018; Mehos and Ramirez, 1989; Van Doren et al., 2011). In this experiment, one assumption is accurate measurement of reservoir petrophysical properties, which are in theory uncertain. Examples of such uncertain parameters are porosity, permeability, fault transmissibilities, etc. Their inaccurate measurements are due to limited reservoir observations and the complex nature of the subsurface flow. The main work of data assimilation (Jung et al., 2018; Nævdal et al., 2006), a compartment in the closed-loop reservoir management, revolves around finding suitable estimates for these parameters.

The closed-loop framework (see Figure 1.3) refers to a process that combines model-based optimization and computer-assisted history matching or data assimilation (see Jansen et al. (2009) for detailed description). It is known to be the most effective way to exploit limited oil reserves more efficiently and economically (Hou et al., 2015). In Figure 1.3, the reservoir, wells, and facilities constitute the physical system (virtual asset) in the closed-loop framework. The system models are realizations of a prior distribution for the physical system, and each may include the static (geological), dynamic, and wellbore flow models. Because it is not possible to accurately quantify the physical system parameters, the system models are known to contain uncertain parameters. Therefore, the reason for the arrows called "Noise", which simply means the input (well control) data (on the left) and the predicted or simulated data (called "Output") contain some errors. The sensors help to keep track and get information (like production data) about the processes in the system. The red loop at the bottom is called the data assimilation process, while the blue loop on the left is called the model-based reservoir optimization.

In reservoir management, the uncertain parameters are quantified by selecting an ensemble of geological realizations of the reservoir of interest. Therefore, knowing that the effects of any EOR methods depend mainly on the reservoir petrophysical properties, it is valuable to consider reservoir uncertainty descriptions for their accurate economic predictions. Therein, one considers optimizing the expectation of a given reservoir performance over all the given geological realizations. It is often called robust optimization. The need to determine the actual benefit of optimal EOR strategies and their ranking with respect to water flooding in the face of geological uncertainty is one reason for this study.

Conventionally, the process of finding the best injection strategy for an oil recovery method (e.g., polymer flooding) in the development of a given reservoir is done by manually selecting a set of upscaled injection strategies based on experiment and

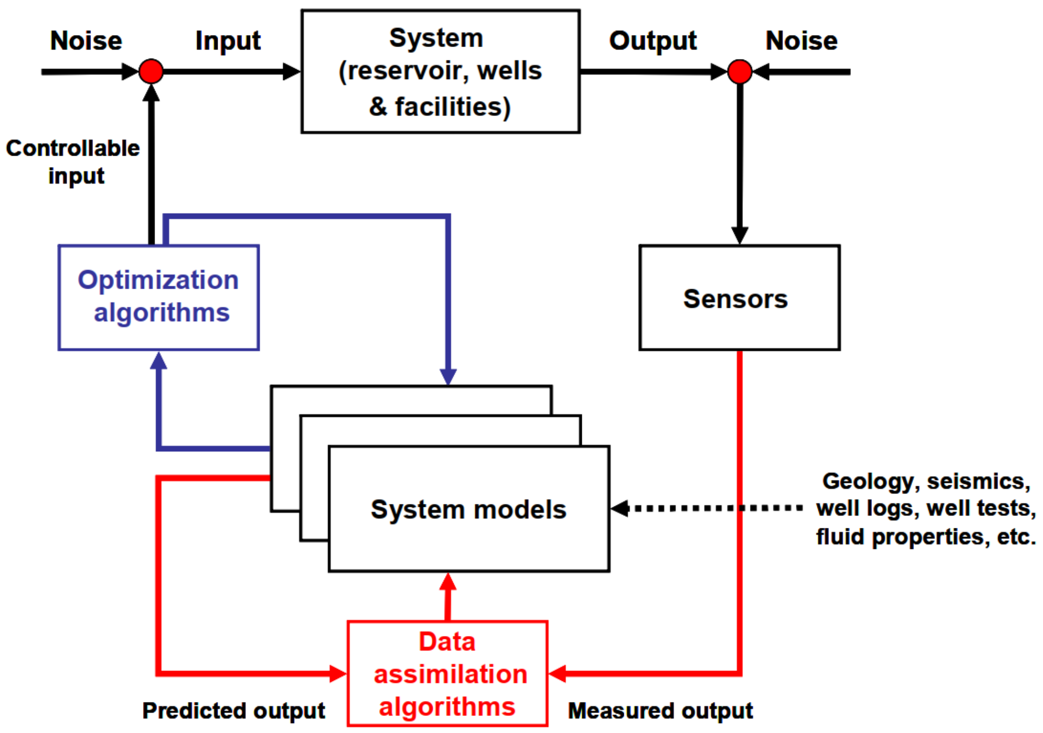

Noise
Input
System (reservoir, wells & facilities)
Output
Noise
Controllable input
Optimization algorithms
Sensors
System models
Geology, seismics, well logs, well tests, fluid properties, etc.
Data assimilation algorithms
Predicted output
Measured output

Figure 1.3: Elements of closed-loop reservoir management (Jansen et al., 2009).

then applying each strategy to the reservoir model to evaluate its economic benefit or profitability, see, e.g., Alfazazi et al. (2019). It is time-consuming and usually automated by formulating the process as a constrained optimization problem. In the reservoir engineering community, this is called production optimization.

The commonly used solution method for this problem is the EnOpt method because it takes into account the uncertainty description in the reservoir (Chen et al., 2009; Chen and Oliver, 2010). However, the current constraint handling techniques utilized with this method possess additional uncertainty in the optimization result. In the case of bound constraints, it is common to use linear or logarithmic projection such as in (Li and Reynolds, 2011; Chen et al., 2009; Do and Reynolds, 2013) to enforce these constraints on the unknown variables at each optimization iteration. This technique is not easily applicable for complicated non-linear constraints, and it can affect solution method accuracy and efficiency due to the impact on gradient quality. Therefore, it is crucial to seek an improved methodology to deal with constraints when applying the EnOpt method for this class of constrained problems to ensure improved performance of the solution method.

In reservoir production optimization for EOR methods, the objective function (or the input-output map) to optimize is usually some reservoir performance measure (such as the NPV function) defined on a given set of controllable EOR variables (such as the water rate, EOR gas rate or chemical concentration of each injecting well at each control time step, etc.) through reservoir simulation (Lee and Aronofsky, 1958; Van Doren et al., 2011; Lei et al., 2012; Dudek et al., 2021). Reservoir simulation for a given EOR method is a dynamic process that predicts the future performance (in terms of oil production, water cut, etc.) of an oil reservoir with the EOR flooding. It involves solving a coupled system of complex time-dependent non-linear partial differential equations describing the behavior EOR gas or chemical flow in the actual oil reservoir on a given grid system (Rasmussen et al., 2021; Bao et al., 2017; Schlumberger, 2010). Grid block sizes in a reservoir simulation model are usually in the order of tens to hundreds of meters in the directions perpendicular to the geological layers, and the model may contain tens of thousands up to millions of grid blocks. Typical simulation time steps are in the other of weeks to months, and a single reservoir simulation over the producing life span of an actual field takes hours to days of runtime (Jansen et al., 2008). In the case of a large oil reservoir field, this simulation requires a lot of computation effort to complete.

Each evaluation of the objective function at a given set of EOR control variables requires performing a complete reservoir simulation. In practice, a better approximation of the objective function gradient at a given EOR control vector by the EnOpt method requires at least 100 different reservoir simulations at each optimization iteration. It makes the solution method very inefficient to use for field applications. Majorly for water flooding modeling, several works have been done to reduce the cost of reservoir simulation. In this case, the general idea involves building a surrogate model for the reservoir simulation using Reduced Order Model (ROM) techniques (Cardoso and Durlofsky, 2010; Van Doren et al., 2006; Sun and Xu, 2017; Durlofsky, 2010). Although this method proves to be modest computation-wise and produces accurate results, it is hard to use with commercial reservoir simulators because its numerical implementation requires access to the discretized underlying mathematical flow equations. It means they are intrusive. Moreover, since most ROM techniques are formulated using proper orthogonal decomposition (POD) and/ or Galerkin methods, they tend to suffer from non-linear inefficiency and instability issues (Walton et al., 2013; Schlegel and Noack, 2015).

Because of above mentioned reasons, many researches have sought non-intrusive alternatives like machine learning (ML) based ROM techniques (Yang and Wang, 2021; Zhong et al., 2020; Shirangi, 2012; Milk et al., 2016). Here, no complicated calculations with the underlying flow physics are required and, by practice, proven to provide good approximating models. However, most of these ML methodologies depend heavily on the hyper-parameter values and quality of the training data. Tuning of hyper-parameters for the sake of finding a good model approximation can be time-consuming. One way to solve this problem is by incorporating a good certifying technique for the ML algorithm.

Also, approximating the full reservoir simulator using non-intrusive methods requires large data set of different features quantifying reservoir states in each grid block and geological properties. There is a possibility to treat the reservoir model completely as a black-box and only consider to approximate the input-output objective function. The aim to find a suitable and certified technology that adaptively approximates the computationally demanding input-output map when using the EnOpt method for EOR optimization problems to have an efficient and desirable accurate decision-making tool is the final motivation for this study.

1.3 Research objectives

The main goal of this PhD project is to develop a robust, accurate, and efficient optimal solution method for constrained production optimization problems associated with the commonly utilized EOR methods for field applications and evaluate actual economic benefits of EOR strategies relative to the traditional oil recovery method (i.e., water flooding). This study achieves its goal by the following three objectives;

- Develop methodology and tools to allow for mathematically well founded optimization and quantification of EOR methods such as polymer, Smart water and CO_2 for relevant North-Sea cases. Hence, on a large scale, provides an extensive and improved understanding towards evaluating reservoir performance for EOR models.

- Develop and implement an efficient and accurate constrained-handling technique for ensemble-based solution methods used for robust constrained (linear or non-linear) reservoir production optimization problems.

- Develop a non-intrusive, efficient, and certified ROM techniques for the computationally expensive objective function (NPV) in the EOR optimization problem and evaluate improvements in the computational speed and NPV approximation respectively.

1.4 Main contributions

In the course of researching solutions to the identified impending problems faced with during reservoir management, and based on the outlined research objectives mentioned above, several papers are published. The research results therein are summarized as follows:

Paper 1: On the Robust Value Quantification of Polymer EOR Injection Strategies for Better Decision Making. This paper presents mathematical tools for optimizing and quantifying the value (with respect to the conventional oil recovery method) of polymer EOR control strategies. The developed methodology is demonstrated with synthetic oil reservoirs with different characterizations. The purpose of the work is to improve the understanding of the actual benefit of polymer flooding and to provide a methodology that quickly allows users to find optimal injection and production strategies that maximize the annually discounted economic values of the injected and production data, the NPV. The polymer control prediction problem is formulated as a constrained optimization problem and the unknowns include polymer concentration, water, oil rates or bottom hole pressures. To account for the uncertainty in the reservoir, an ensemble of geological realizations is used. An EnOpt method with covariance adaptation is utilized to solve the optimization problem. Important findings of this study are the feasible control strategies for polymer EOR methods leading to an increased NPV, and the observed difference of the economic values for polymer and traditional water flooding for the examples considered.

Paper 2: Ensemble-based Constrained Optimization Using Exterior Penalty Method. This paper proposes a new efficient, robust, and accurate optimal solution strategy based on the Exterior Penalty Function (EPF) method and the adaptive EnOpt approach (with backtracking line-search technique) for non-linear constrained optimization problems. This work provides a better user-friendly strategy which mitigates the problem often faced with the current constraints handling technique utilized when using the EnOpt method to solve constrained problems of water or EOR flooding. It is noted that the problem contributes to uncertainties in the gradient computation of the objective function and hence leads to the poor convergence rate of the standard acenopt method. Here, the EPF method is used to transform a given constrained optimization problem to a sequence of unconstrained subproblems and then sequentially solve the subproblems by unconstrained EnOpt procedure until convergence to the solution of the original problem. The demonstration of proposed methodology with analytical constrained problem and practical high dimensional bound constrained water flooding optimization problem associated with a 2D and a 3D reservoir fields show a fast convergence and more accurate results than traditional method.

Paper 3: Robust Value Quantification of EOR Injection Strategies. This paper evaluates the economic benefits and environmental impacts of the optimal control strategies for the commonly utilized EOR methods such as polymer, Smart water, and CO_2 flooding compared to the conventional oil recovery method. Knowing that the aceor effects of the recovery methods are sensitive to fluid and rock properties in the reservoirs, for appropriate prediction, the uncertainties in the reservoirs are considered by using an ensemble of geological realizations obtained by engineering upscaling of the initial model. In the optimization problems, the unknowns are EOR gas rate or chemical concentration, water rates, oil rate, or bottom hole pressures. Also, the effect of different injection costs of CO2 on the optimization results of CO2 EOR method is investigated.

Paper 4: Adaptive machine learning based surrogate modeling to accelerate PDE-constrained optimization for enhanced oil recovery. This paper proposes a novel certified machine-learning-based ROM method for different versions of the input-output objective function in EOR optimization problems and surrogate model adaptation technique during the optimization procedures. This work demonstrates that the computational cost of reservoir simulations required during optimization renders the traditional method like the EnOpt for solving the optimization problems very inefficient in making decision. To reduce this computational effort without compromising solution accuracy, we propose to approximate the non-linear objective function, which depends heavily on reservoir simulation using feedforward deep neural networks. The ROM method, combined with the EnOpt method is used to solve a polymer flooding optimization problem on a

5Spot field. In this problem, we aim to find the best control strategy, including polymer concentration and water rates for the injection wells and oil rates for the production wells over the reservoir production lifespan that gives maximum net present value as fast as possible. The proposed method is found to be more efficient, and it gives an improved solution to the polymer optimization problem than the traditional solution method.

1.4.1 Study outlined

This study provides a detailed introduction based on scientific background of the key subject matter to better understand the scientific results in the above-mentioned papers. It is subdivided into two parts, namely Part I; the scientific background and Part II; the scientific contributions of study.

The rest of Part I is arranged as follows; Chapter 2 demonstrates the physical interpretation and mathematical formulation of the black-oil model and its extensions describing polymer, Smart water, and CO_2 EOR flooding for appropriate NPV formulation. Based on the results in Chapter 2, the general constrained optimization problem associated with EOR methods is presented in Chapter 3. Chapter 4 looks at the general procedures suitable for approximating the gradient of the EOR NPV function for deterministic and robust settings and, also, the solution method to solve the underlying constrained EOR optimization problem. Chapter 5 illustrates the application of model order reduction techniques in reservoir optimization problems and Chapter 6 draws a general conclusion from the studies.

Part II presents the scientific contributions and results of this PhD study in the form of published papers.

2 EOR modeling and open porous media

This study considers optimizing the field development of oil reservoirs of interest with three EOR methods, namely, polymer, Smart water, and CO_2 or solvent flooding. For convenience, polymer and Smart water are grouped as chemical EOR methods and CO_2 as a gas EOR method. The optimization process involves an objective function defined based on some controllable variables. In this study, the NPV function is considered as the objective. It quantifies the economic value of the injected and produced fluids over a given period of time in the reservoir. To every fluid injected into the reservoir, there is a need to appropriately measure the reservoir response (or performance) in terms of the produced fluids which depends on state of the reservoir at a point. An important technique for predicting reservoir performance under a given operating condition is the reservoir simulation. For the formulation of EOR optimization problems, the mathematical models for simulating the recovery effect of the EOR methods of interest are first described. The EOR models, which are extensions of the most widely used black-oil model for fluid flow simulation in oil reservoirs are utilized in this study.

2.1 Black-oil model

The black-oil model consists of a set of partial differential equations (PDEs) governing the simultaneous flow behavior of two or three different phases, namely, water (aqueous), oil (oleic), and gas (gaseous) phases and pseudo components, namely, water, oil, and gas in an hydrocarbon reservoir. The term "black-oil" is due to the assumption that all the hydrocarbon species can co-exit as gas and oil with constant chemical composition at surface conditions. There is no mass transfer between the fluid components but mixing is possible. That is, both oil and gas can partially or completely dissolve in each other to form, depending on the pressure, a oleic or gaseous phase at reservoir conditions (Bao et al., 2017).

The model equations are formulated from conservation of mass for each component coupled with suitable closure relations, initial, and boundary conditions. Let $\Omega \subset \mathbb{R}^3$ be the spatial domain of a reservoir with boundary $\partial\Omega \subset \mathbb{R}^3$ and $T = \{t \in \mathbb{R} : t \geq 0\}$ be the time domain with boundary $\partial T = \{t \in \mathbb{R} : t = 0\}$. For simplicity, quantities associated with the aqueous, oleic, and gaseous phases are identified with subscripts, $w, o,$ and g respectively. Let $A_\alpha := A_\alpha(\mathbf{s}, t)$ be the α phase accumulation as function of position $\mathbf{s} := (s_1, s_2, s_3) \in \Omega$ and time

$t \in T$.

2.1.1 Balance equation

For fluid phase (or pseudo-component) $\alpha \in \{w, o, g\}$, the conservation of mass (or balance) equation is given by (note that in this chapter, bold small and capital letters indicate vector and tensor quantity respectively):

$$\frac{\partial}{\partial t}(\phi \rho_\alpha A_\alpha) + \nabla \cdot (\rho_\alpha \boldsymbol{\tau}_\alpha) = Q_\alpha, \quad \mathbf{s} \in \Omega \cap \partial\Omega^c, \ t \in T \cap \partial T^c \tag{2.1}$$

(superscript c indicates set complement). In this case, $\boldsymbol{\tau}_\alpha = \boldsymbol{\tau}_\alpha(\mathbf{s}, t)$ is the $\alpha-$phase surface volume flux through the porous medium. It is a 3-dimensional vector, which describes the phase flow along each spatial coordinate axis. The quantity $Q_\alpha = Q_\alpha(\mathbf{s}, t)$ denotes the source/sink term for the $\alpha-$phase. It models the flow of the phase from the injection wells into the reservoir and flow to the production wells (see Appendix A.1 and **Paper IV** for example of well model). ϕ denotes the porosity of the reservoir rock formation. The porosity is the volumetric fraction of the reservoir rock that is void space. The quantity ρ_α is the density of $\alpha-$phase.

2.1.2 Constitutive equations

The equations for modeling reservoir fluid and rock parameters in the balance equation given by (2.1) are presented as follows.

The accumulation term for each $\alpha-$phase is given respectively by:

$$A_w = b_w S_w, \tag{2.2}$$
$$A_o = b_o S_o + r_{og} b_g S_g, \tag{2.3}$$
$$A_g = b_g S_g + r_{go} b_o S_o, \tag{2.4}$$

where b_α is the inverse of volume formation factor of alpha phase. The volume formation factor describes the ratio of volume (at reservoir condition) of a phase to the volume (at surface condition) of the phase. $S_\alpha = S_\alpha(\mathbf{s}, t)$ is the saturation of the $\alpha-$phase. It measures the fraction of pore space that the $\alpha-$phase occupies. For three phase flow of oil, water, and gas, the sum of their saturations gives 1. This implies that the three phases jointly occupy the pore spaces.

In this regards, the surface volume fluxes of the three phases are computed by

$$\boldsymbol{\tau}_w = b_w \mathbf{v}_w, \tag{2.5}$$

$$\boldsymbol{\tau}_o = b_o \mathbf{v}_o + r_{og} b_g \mathbf{v}_g, \tag{2.6}$$

$$\boldsymbol{\tau}_g = b_g \mathbf{v}_g + r_{go} b_o \mathbf{v}_o, \tag{2.7}$$

where r_{go} measures the volume of gas (measured at standard conditions) dissolved at a given reservoir pressure and temperature in a unit volume of oil. Similarly, r_{og} denotes the volume of oil dissolved in a unit volume of gas. The quantity $\mathbf{v}_\alpha = \mathbf{v}_\alpha(\mathbf{s}, t)$ denotes the velocity at which the $\alpha-$phase flows through the porous medium and is modeled by Darcy's law as

$$\mathbf{v}_\alpha = -\frac{k_{r\alpha}}{\mu_\alpha} \mathbf{K}(\nabla p_\alpha - \rho_\alpha \mathbf{a} \nabla s_3). \tag{2.8}$$

Here, $p_\alpha, k_{r\alpha}$, and μ_α denote the pressure (and ∇p_α is the pressure gradient), relative permeability[1], and viscosity[2] of phase α respectively. Furthermore, $\mathbf{a}$ is the gravitational acceleration vector, $\mathbf{K}$ is the permeability[3] tensor of the porous medium, and ∇s_3 is the change in the s_3-direction.

Reservoir parameter modeling

Good understanding of the mathematical formulations regarding quantities involving reservoir rock, fluid, and rock-fluid interaction properties for the black-oil model considered in this study is documented in (Chen, 2007). The reservoir rock properties of interest measuring the capacity of reservoir rock to transmit and store fluids in its pores include porosity and permeability.

Fluid properties of interest including densities, viscosities, formation volume factors, gas solubility factor, etc., are assumed to depend on phase pressure and rock-fluid interaction properties. The rock-fluid interaction property such as capillary pressures and relative permeabilities depend on phase saturation.

[1]The relative permeability of a phase measures the amount of impairment to flow of the phase on another. In two-phase flow, it depends on the phase saturation; in three-phase flow, each phase relative permeability can depend on more than one phase saturation.

[2]The viscosity of a phase measures the internal resistance force per unit surface area due to an applied shearing force in the opposite direction of the phase flow.

[3]The rock permeability measures the capacity of the rock to conduct fluids through its interconnected pores.

2.1.3 Reservoir state variables

Combining Equations (2.1) - (2.8) leads to a system of PDEs with primary state variables (unknowns), p_o, p_w, p_g, S_o, S_g, and S_w. They are called state variables because they change with time. In the case of non-miscible flow (such as in water flooding, etc), the resulting coupled PDEs primary unknowns are reduced to, p_o, S_w, and S_g using the following closure properties:

$$
\begin{aligned}
S_w + S_o + S_g &= 1, && (2.9)\\
p_{c,ow}(S_w) &= p_o - p_w, && (2.10)\\
p_{c,og}(S_g) &= p_o - p_g, && (2.11)\\
p_{c,gw}(S_w) &= p_g - p_w = p_{c,ow}(S_w) + p_{c,og}(S_g), && (2.12)
\end{aligned}
$$

and appropriate state equations [4]. Here, $p_{c,ow}$, $p_{c,ow}$, and $p_{c,gw}$ denote oil-water, oil-gas, and gas-water saturation dependent capillary pressures[5] respectively. In the rest of this study, the state vector at a given time t will be denoted as $\mathbf{x}(t)$.

In the miscible flow context (like solvent flooding), the set of primary unknowns is different because the gaseous phase vanishes when all the gas completely dissolves into the oleic phase. Likewise, the oleic phase could disappear when oil vaporizes into the gaseous phase. Therefore, using S_g as the third primary variable is not suitable. In this regards, the third unknown is defined to keep track of the phase composition present locally in each grid cell. In this case, when all the three phases are present, it is S_g, when no gaseous phase is present, it is r_{go}, and if no oleic phase is present, it is r_{og}.

2.1.4 Initial and boundary conditions

Additional data, the initial and boundary conditions are imposed on the resulting coupled PDEs (from (2.1) - (2.12)) to give a complete approximately solvable black-oil model. In general, the initial $\alpha-$phase accumulation in the reservoir is

[4]State equations are thermodynamic equations relating state variables, which describe the state of matter under a given set of physical conditions, such as pressure, volume, temperature, or internal energy, e.g., $c_o = \frac{1}{\rho_o}\frac{\partial \rho_o}{\partial p_o}|_T$. Here, c_o is the oil compressibility factor at constant temperature T (Neumann et al., 2010).

[5]In two-phase flow, a discontinuity in pressure occurs across an interface between any two immiscible fluids due to the presence of interfacial tension. This discontinuity in pressure is called capillary pressure.

expressed as

$$A_\alpha(\mathbf{s}, t) = A_\alpha(\mathbf{s}), \quad \mathbf{s} \in \Omega,\ t \in \partial T. \tag{2.13}$$

Since each accumulation term is a function of the state variables, specificially, the initial data can be prescribed directly for the unknown state vector $\mathbf{x} = [p_o, S_w, x_3]$, with

$$x_3 = \begin{cases} S_g, & \text{oleic, gaseous, and aqueous phases present} \\ r_{go}, & \text{no gaseous phase} \\ r_{og}, & \text{no oleic phase} \end{cases}. \tag{2.14}$$

For example, the hydrostatic pressure and fluid distribution data can be given as initial conditions. The boundary condition usually involves data specifying no fluid interchange with the surrounding reservoir rock formation, that's

$$\boldsymbol{\tau}_\alpha(\mathbf{s}, t) = 0, \quad \mathbf{s} \in \partial\Omega, t \in T, \tag{2.15}$$

except through the well models (no-flow Neumann conditions). Other approaches of specifying the initial and boundary data can be found in (Rasmussen et al., 2021). This gives a closed system of PDEs called the black-oil model equations (BOMEs).

2.1.5 Numerical solution of BOMEs

The solution strategy starts with suitable discretization toolbox to appropriately discretize the differentials in BOMEs simultaneously with respect to space and time, and the resulting system of non-linear equations are solved using suitable non-linear or linear (coupled with linearization technique) solver.

In this study, the BOMEs are discretized in space into a suitable number of grid blocks using two point flux approximation (TPFA) with upstream-mobility weighting (UMW) to give a set of ordinary differential equations (ODEs)

$$\mathbf{g}(\mathbf{u}, \dot{\mathbf{x}}, \mathbf{x}, \boldsymbol{\theta}) = \mathbf{0}. \tag{2.16}$$

Here, $\mathbf{g}$ is a non-linear vector-valued function representing the black-oil simulator, $\mathbf{u}$ is an arbitrary input vector of well controls such as well flow rates (or outflux density), well bore pressure (either in the reservoir or at the surface), valve settings in the gridblock penetrated by wells, etc., $\mathbf{x}$ is the state vector, and $\dot{\mathbf{x}}$ is the first derivative of the state vector components with time, and $\boldsymbol{\theta}$ is a reservoir model parameter vector containing parameters such as porosities, permeabilities, and other static reservoir or fluid properties.

On the application of first-order fully implicit temporal discretization (FOFITD), the ODEs (2.16) gives a system of non-linear discrete-time equations

$$\mathbf{g}_{i+1}(\mathbf{u}_{i+1}, \mathbf{x}_{i+1}, \mathbf{x}_i, \boldsymbol{\theta}) = \mathbf{0}, \qquad i = 0, 1, 2, ..., N_t - 1, \tag{2.17}$$

$$\mathbf{x}_0 = \overline{\mathbf{x}}_0 \tag{2.18}$$

where the subscript i denotes time step index and N_t is the simulation time. In Equation (2.17), $\mathbf{x}_i$ is the shorthand for $\mathbf{x}(t_i)$ and it implies the state vector value at time $t = t_i$. Further, $\overline{\mathbf{x}}_0$ is the prescribed initial data. At each time step, the system (2.17) - (2.18) is solved iteratively using Newton-Raphson (NR) method given by

$$\mathbf{x}_{i+1,n+1} = \mathbf{x}_{i+1,n} - \mathbf{F}_{i+1,n}[\mathbf{F}'_{i+1,n}]^{-1}. \tag{2.19}$$

Here, $n = 0, 1, ...,$ is the NR iteration index, $\mathbf{F}_{i+1,n}$ and $\mathbf{F}'_{i+1,n}$ denote the value and Jacobian of $\mathbf{g}_{i+1}$ at $\mathbf{x}_{i+1}$ in the $n-$th NR-iteration respectively. The process continues until (2.17) satisfies a prescribed convergence criteria such as

$$||\mathbf{g}|| \leq \epsilon_x, \tag{2.20}$$

where $\epsilon_x > 0$ (very small) is a given tolerance. At each time step, the outputs (such as oil rate, water production, etc.) from the production wells depends on the approximate solution from (2.19) and the input vector. These information are vital for the computation of the NPV of the reservoir. Therefore, for convenience, the output vectors of interest at each time step is denoted by:

$$\mathbf{y}_i = \mathbf{h}_{\boldsymbol{\theta},i}(\mathbf{u}_i, \hat{\mathbf{x}}_i), \qquad i = 1, 2, ..., N_t \tag{2.21}$$

where $\hat{\mathbf{x}}_i$ is the solution from (2.19) at the ith time step. The process of obtaining (2.21) is often called forward reservoir simulation.

The black-oil model demonstrated in this section is useful in simulating water flooding (WF) and water alternating gas (WAG) flooding in oil reservoir. A robust research purpose black-oil simulator used in this study is the Open Porus Media (OPM) Flow (Baxendale et al., 2021). It is an open-source alternative for the popularly known commercial simulators and can be used to implement, test, and validate new reservoir models and computational methods in realistic and industrial settings. For an extensive comparison studies of the OPM Flow and commercial simulators, the reader is referred to the work of (Rasmussen et al., 2021).

2.2 EOR model

The above-explained black-oil model has been used extensively to predict the performance of oil reservoirs under operating conditions such as water or WAG flooding for reservoir evaluation and optimization. However, for operating conditions such as EOR flooding, the model becomes unrealistic. The subsequent subsections present extensions of the black-oil model suitable for simulating reservoir performance under EOR flooding conditions of interest to this study.

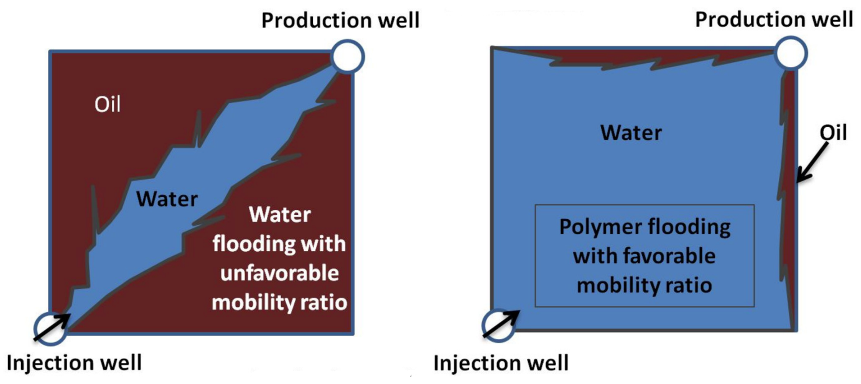
2.2.1 Polymer model

In an oil reservoir with an unfavorable mobility ratio, water flooding experiences viscous fingering, as shown in Figure 2.1a. This affects the displacement mechanism of water flooding resulting in a high volume of left-over oil. In this scenario, polymer flooding is one of the most widely used chemical EOR methods known to mainly improve the mobility ratio in a water flooding setting, causing an improved sweep efficiency. Injecting polymer into water flooding helps to reduce the occurrence of viscous fingering effect by increasing the viscosity of the injected fluid (equivalently decreasing mobility ratio of oil-water system) and, consequently, improves the sweep efficiency of water flooding, as shown in Figure 2.1b.

For simulating polymer flooding, this study considers the black-oil model in Section 2.1, with additional continuity equation for polymer component assumed to be transported in the water phase and has no effect on the oleic phase. The resulting model is called the polymer EOR model. In addition, the model accounts for the effect of polymer on dead pore space, adsorption in the rock, and permeability reduction. Therefore, the polymer model is obtained by the inclusion of the following continuity equation in the black-oil model.

$$\frac{\partial}{\partial t}\left[\phi\rho_w(1-S_{ipv})S_w + \frac{\rho_r c^a}{b_w c}(1-\phi)\right] + \nabla\cdot(\rho_w \mathbf{v}_p) = Q_w, \tag{2.22}$$

where $0 \leq c \leq 1$ is the polymer concentration expressed as the mass per unit volume of water, $c^a := f(c)$ is a function of c quantifying the polymer adsorption


Production well
Oil
Water
Water flooding with unfavorable mobility ratio
Injection well
Production well
Water
Oil
Polymer flooding with favorable mobility ratio
Injection well

(a) Fingering effect (b) Non-Fingering effect

Figure 2.1: Fingering effect promoted by the unfavorable mobility ratio (left), and good oil recovery facilitated by the use of polymer flooding (right) (Zerkalov, 2015).

concentration, ρ_r is the reservoir rock density, and S_{ipv} is the fraction of the reservoir pore volume that is inaccessible. Further, $\mathbf{v}_p$ is the polymer flux rate given by

$$\mathbf{v}_p = -\frac{k_{rw}}{\mu_{p,\mathrm{eff}} R_k}\mathbf{K}(\nabla p_w - \rho_w \mathbf{a} \nabla s_3), \tag{2.23}$$

where $\mu_{p,\mathrm{eff}}$ is the effective polymer viscosity and R_k models the permeability reduction effect due to polymer adsorption onto the reservoir rock surface.

Since the mechanism of polymer flooding is to increase the water viscosity, the flux rate of the water phase computed using Equation 2.8 is modified based on the effective water viscosity denoted by $\mu_{w,\mathrm{eff}}$. The new flux rate becomes:

$$\mathbf{v}_w = -\frac{k_{rw}}{\mu_{w,\mathrm{eff}} R_k}\mathbf{K}(\nabla p_w - \rho_w \mathbf{a} \nabla s_3). \tag{2.24}$$

To compute the effective viscosities $\mu_{p,\mathrm{eff}}$ and $\mu_{w,\mathrm{eff}}$, the Todd-Longstaff model in (Todd and Longstaff, 1972) is used. Here, the degree of polymer mixing with water is measured by a mixing parameter $\omega \in [0, 1]$. Usually, the selection of ω is based on the type of displacement setting, the geological heterogeneity, etc. of the reservoir. When $\omega = 1$, it means polymer is fully mixed with water and $\omega = 0$ implies there is complete separation between polymer solution and pure water. Assume that μ_{fmp} denote the viscosity of a fully mixed polymer with water solution, the effective polymer viscosity is computed as

$$\mu_{p,\text{eff}} = \mu_{\text{fmp}}^{\omega} \cdot \mu_{p}^{1-\omega}, \tag{2.25}$$

where $\mu_p = \mu_{\text{fmp}}(c^*)$ is the viscosity of fully mixed polymer solution containing the maximum allowable polymer concentration c^*. Similarly, the viscosity of partially mixed water calculated as

$$\mu_{\text{pmw}} = \mu_{\text{fmp}}^{\omega} \cdot \mu_{w}^{1-\omega}. \tag{2.26}$$

Using Equations (2.25) and (2.26), the effective water viscosity is calculated by

$$\frac{1}{\mu_{w,\text{eff}}} = \frac{1-\overline{c}}{\mu_{\text{pmw}}} + \frac{\overline{c}}{\mu_{p,\text{eff}}}, \tag{2.27}$$

where $\overline{c} = \frac{c}{c^*}$. Experimentally, it has been shown that polymer may stick to the rock surface by adsorption process, which in turn reduces the polymer concentration and hence causes resistance to flow which decreases the effective permeability of water. For this reason, the accumulation term $\rho_r c^a (1 - \phi)$ is introduced in Equation (2.23), to model this instantaneous and reversible process.

In general, the polymer model described in this section is known as a 4-component multiphase black-oil flow model. Since the polymer component is transported in the water or aqueous phase and has no effect on other phases (or pseudo-components) except the water component, the primary state variables remains as explained for the general black-oil model (see, Section 2.1). Therefore, solution strategies for the polymer model equations also follows from the demonstration in Sub-section 2.1.5 and it is fully implemented in the OPM reservoir simulator.

Further, the input vector **u** in Equation (2.16) also includes polymer concentration (and hence polymer rate) at the injection wells and the output vector in (2.21) includes polymer production rate at the production wells.

2.2.2 Smart water model

Smart water (otherwise called low salinity water) flooding is a quite new EOR method compared to polymer method, currently receiving huge treatise because of its cost-effective, low environmental impact, and potential of increasing oil recovery (Yousef et al., 2011). This type of flooding involves the injection of chemistry-optimized water in terms of salinity (or salt concentration) and ionic composition (Yousef et al., 2012) into the reservoir.

Reservoir wettability

Wettability is a reservoir rock surface property, which measures the preference of the rock surface to be wetted by a given phase or fluid and can affect the fluid displacement process (Chen, 2007). For instance, a water wet rock implies that the preferred fluid by the rock surface is water. Both capillary and relative permeability effects are influenced by the wetting behavior of the rock in which the oil is found.

If the rock surface is water wet (see, Figure 2.2a) then there is a tendency to have lesser residual oil saturation (the proportion of oil which remains permanently trapped by capillary effects at the pore scale). This is caused by the growth in the water film on the rock surface during water flooding, which ultimately leads to water squeezing out trapped droplets of oil within the pores, see Figure 2.2b. Consequently, more oil is produced at the production well as water flooding continues until the breakthrough point, see Figure 2.2c.

If the rock is oil wet, then the proportion of oil trapped by capillary effects is much lower, as oil continuity is maintained over the rock surfaces and through the pore throats, but water breakthrough is earlier and there is a long period of time during which oil and water are produced simultaneously. The net result is that overall recovery is generally higher if the reservoir rock is oil wet but only after a very large throughput of water.

Most oil reservoir rocks are thought to have a heterogeneous wettability, usually termed “mixed wettability”, in that larger pores and throats have both water- and oil-wet surfaces but smaller pores remain mainly water wet Muggeridge et al. (2014). It is believed that the reservoir rock changes from an initially water-wet state to this mixed wettability state after the migration of oil into the reservoir.

The main EOR mechanism of Smart water is due to the chemistry between its composition and the reservoir rock surface causing favorable changes in rock wettability. Here, an initially oil-wet and/ or intermediate-wet rock surfaces become more water-wet due to contact with Smart water. Consequently, more oil becomes detached from these surfaces to increase oil production (Fani et al., 2018; Ridwan et al., 2020; Yousef and Ayirala, 2014).

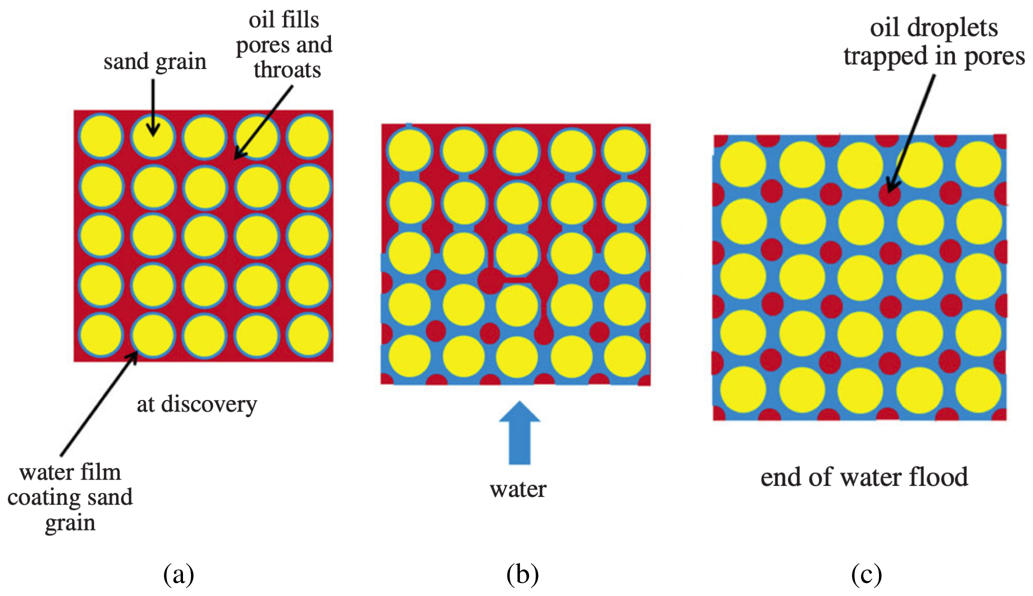

sand grain
oil fills pores and throats
at discovery
water film coating sand grain
water
oil droplets trapped in pores
end of water flood
(a)
(b)
(c)

Figure 2.2: Illustration of oil trapping in a water-wet rock. (a) At discovery the sand grains are coated with a thin water film and the pores are filled with oil; (b) as water flooding progresses the water films become thicker until; (c) the water films join and oil continuity is lost (Muggeridge et al., 2014).

This study considers the salinity aspect of Smart water while the ionic composition remains fixed and assumes there is only one salt specie. To simulate Smart water flooding, similar assumptions for polymer are utilized. It is transported in the water phase as additional component in the black-oil model of Section 2.1. In addition, it can modify properties of other phase components to increase oil production. For instance, a given salinity can cause changes in the water density and viscosity, the saturation and relative permeability end-points of oil and water, and water-oil capillary pressure.

The Smart water model considered in this study includes the Black-oil equations in Section 2.1 and the continuity equation for salinity given in compact form as:

$$\frac{\partial}{\partial t}\left(\frac{\phi \rho_w S_w c_s}{b_w}\right) - \nabla \cdot \left(\frac{\rho_w k_{rw}}{b_w \mu_{s,\text{eff}}}\mathbf{K}(\nabla p_w - \rho_w \mathbf{a} \nabla s_3)\right) c_s = Q_w c_s. \tag{2.28}$$

Here, c_s is the salinity of Smart water and $\mu_{s,\text{eff}}$ is the effective viscosity of salt and it is calculated using the similar Todd-Longstaff procedure for polymer model. Other parameters remains as before.

The reasoning regarding type of and solution for state variables in the Smart water model is analogous to the polymer model described in the previous Section. In addition, the input vector **u** in Equation (2.16) include Smart water concentration (and hence Smart water rate) at the injection wells and the output vector in (2.21) includes Smart water rate at the production wells.

2.2.3 Solvent model

So far, the chemical EOR models discussed in two sub-sections above are for immiscible flooding. Here, the EOR chemicals such as polymer and smart water are not miscible with the oil to form homogeneous mixture in order to improve total oil recovery.

This section presents the mathematical modeling of solvent or carbon dioxide (CO_2) EOR method that is capable of becoming miscible with the oil at a relatively high pressure called the Minimum Miscible Pressure (MMP) (Janiga et al., 2020; Jia et al., 2013). CO_2 is generally soluble in oil at reservoir pressures and temperatures. It diffuses into and swells the net volume of oil in other to reduce its viscosity and the interfacial tension between the oil and oil-CO_2 phase, contributing to microscopic displacement of oil, see Figure 2.3. This process is generally called miscible displacement of oil by CO_2. The injection of CO_2 also affects the relative permeability, residual saturations, and density of reservoir fluids.

Below the MMP, Wang (1980) found that CO_2 undergoes immiscible displacement. More so, the MMP required for CO_2 to become miscible with oil is dependent on several parameters such as slug size and purity of injected CO_2, reservoir temperature, and oil composition (Sehbi et al., 2001).

The modeling approach for solvent flooding usually involves computing the properties for the fully miscible and the immiscible cases and interpolate between the

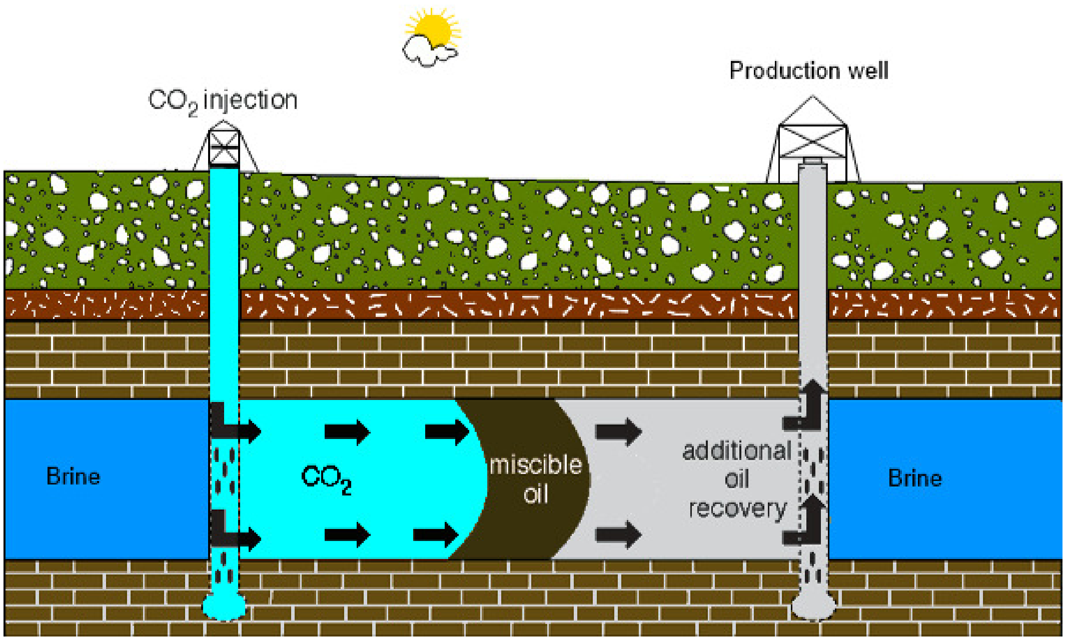


Figure 2.3: Schematic diagram illustrating the use of CO$_2$ injection to Enhance Oil Recovery (Alizadeh Nomeli and Riaz, 2013)

two limits using a measurable function, the miscibilty, dependent of pressure and solvent saturation. The solvent model is formed by extending the black-oil model with a fourth continuity equation for the CO$_2$ component (denoted by s) given by:

$$\frac{\partial}{\partial t}(\phi \rho_s b_s S_s) - \nabla \cdot \left(\frac{\rho_s b_s k_{rs}}{\mu_s} \mathbf{K}(\nabla p_g - \rho_s \mathbf{a} \nabla s_3) \right) = Q_s, \tag{2.29}$$

where Q_s, S_s, k_{rs}, b_s, and μ_s are the outflux density, saturation, relative permeability, inverse of formation factor, and viscosity of solvent. In solvent flooding, it is assumed that in each pore, there can exist four different components namely, oil(o), water(w), solvent(s), and other reservoir gases (g). Therefore, the summation of saturations given in (2.9) is recomputed as;

$$S_o + S_w + S_g + S_s = 1. \tag{2.30}$$

In the usual black-oil model, the relative permeabilities for the three phases namely water, oil, and gas are defined as functions of saturation. That's

$$k_{rw} = k_{rw}(S_w) \tag{2.31}$$

$$k_{rg} = k_{rg}(S_g) \tag{2.32}$$

$$k_{ro} = k_{ro}(S_w, S_g). \tag{2.33}$$

In cases where two gas components (e.g., CO_2(s) and other gas(g)) are present, the total relative permeability k_{rgt} of the gas phase is assumed to be a function of the total gas saturation given as:

$$k_{rgt} = k_{rg}(S_g + S_s). \tag{2.34}$$

The individual relative permeability of solvent and gas is then computed as a function of the fraction of each gas component within the gas phase multiplied by the total relative permeability (2.34). That's

$$k_{rs} = k_{rgt} \cdot k_{rfs}(F_s) \tag{2.35}$$

$$k_{rg} = k_{rgt} \cdot k_{rfg}(F_g). \tag{2.36}$$

Here, $F_s = \frac{S_s}{S_s+S_g}$ and $F_g = \frac{S_g}{S_s+S_g}$ denote the fractions of solvent and reservoir gas components. Typical examples of the relative permeability functions used in the black-oil model and its extensions (such as solvent model, etc.) can be found in (Rasmussen et al., 2021; Schlumberger, 2010).

Since the solvent component also modifies properties such as viscosity and density of hydrocarbon fluids in the reservoir, their respective effective viscosities are calculated using the Todd-Longstaff model (Todd and Longstaff, 1972; Chase and Todd, 1984; Jakupsstovu et al., 2001) and effective densities follow from the work of Chase and Todd (1984).

The present solvent model describes a miscible flooding. In this, there is tendency for the solvent to completely dissolve in the oleic phase forming a homogeneous oil-solvent mixture and vice-versa as explained in the Section 2.1. Therefore, the state variables in the solvent model are the same as (2.14) and their solution strategy follows accordingly as demonstrated in Section 2.1.5. However for this study, the input vector $\mathbf{u}$ in Equation (2.16) include solvent rate instead of water rate at the injection wells and the output vector in (2.21) includes solvent rate at the production wells. The solvent model is also fully supported by the OPM Framework.

The black-oil model and its extensions, the EOR models, have been discussed with three phases, namely oleic, water, and gas. However, for simulations involving two phases such as oleic/water or oleic/gas, the models are simplified by reducing the number of continuity equations accordingly to the phases (or pseudo-components) present, considering appropriate saturation relation and related parametric properties for two-phase flow. In this case, the number of primary state variables to solve for become reduced. Aside from this reduction, other formulations like the wells equation remain the same.

For any reservoir model with geological properties stored as $\boldsymbol{\theta}$, initial dynamic states $\mathbf{x_0}$, and a given well configuration, the simulation of the reservoir response (in terms of production output using (2.21)) with respect to a given EOR flooding control can be done by using the appropriate EOR model present in this chapter.

Different standard units exist for the reservoir parameters considered in this study. In the numerical experiments performed, the metric unit is used. A complete list of metric units of reservoir parameters can be found in (Baxendale et al., 2021; Chen, 2007).

3 EOR optimization

One of the main techniques in reservoir management is mathematical optimization. In general, it is the process of finding the extremum (minimum or maximum) point of an objective function defined on some unknown variables subject to some or no constraints. In the presence of one or more constraints, the process is called constrained optimization. In contrast, it is called unconstrained optimization.

The development goal of a given reservoir usually involves the appropriate selection of control strategy (for the injection and production wells) for a given flooding process in other to increase (maximize) return on investment (ROI) and reduce (minimize) the risk of mismanaging limited resources. This process is formulated as a constrained optimization problem. In this study, the objective function is the NPV function which measures the economic value (or profitability) of all injected and produced fluids discounted from the total cash flow. It is similar to the one in (Xu et al., 2018; Zhou et al., 2013). However in their formulations, there is no account for back-produced EOR chemical/gas.

To measure the quantities of fluids produced due to the response or dynamic change of state (per time) of a given reservoir to injected fluids requires appropriate reservoir modeling to simulate this behavior. The previous chapter presents the general reservoir modeling of EOR methods considered in this study. In this case, the information for the definition of the NPV can be accessed.

Here, the general constrained optimization problem suitable for reservoir development with the EOR methods considered in this study is first discussed, followed by the demonstration of appropriate solution method.

3.1 Constrained EOR optimization problem formulation

The aim of oil reservoir management with a given EOR method is to maximize the economic value of the oil reservoir under limited production and injection facilities. Most importantly, one wants to produce oil as much as possible while the operational cost is at minimum within a given limit.

Suppose that the oil reservoir model of interest has a known geological properties such as porosity and permeability, etc., given by $\boldsymbol{\theta}$. Let $\mathcal{D} \subseteq \mathbb{R}^{N_u}$ be the domain of input control vector

$$\mathbf{U} := \{\mathbf{u}_i\}_{i=1}^{N_t} = [\{u_i^j\}_{i=1}^{N_t}, j = 1, 2, ..., N_w] \tag{3.1}$$

for a given EOR method. Here, N_w is the number of wells in the reservoir, N_t remains its usual meaning as in Section 2.1.5, and $\mathbf{u}_i$ is the input vector of wells at simulation time step i. Further, $N_u = N_w \times N_t$ is the total control variables of each input vector $\mathbf{U}$. For clarity, each component u_i^j of $\mathbf{U}$ denotes a control type such as water rate, EOR chemical/gas concentration or rate, oil rate or bottom hole pressure (for production well) at the $j-$th well in the $i-$th time step. For convenience, the desired outputs from all simulation time steps N_t as a result of $\mathbf{U}$ using Equation (2.21) shall be denoted as

$$\mathbf{Y} = \{\mathbf{y}_i\}_i^{N_t}, \tag{3.2}$$

where $\mathbf{y}_i$ is the output vector at the i-th time step.

The general N_u- dimensional constrained EOR optimization problem is to find the optimal $\mathbf{U} \in \mathcal{D}$ that maximizes the reservoir NPV function subject to bound constraints. That's

$$\underset{\mathbf{U}\in\mathcal{D}}{\text{maximize}}\ J(\mathbf{U}, \boldsymbol{\theta}) := \sum_{i=1}^{N_t} J_i(\mathbf{u}_i, \mathbf{y}_i) \tag{3.3}$$

with

$$J_i(\mathbf{u}_i, \mathbf{y}_i) = \Big[\sum_{j=1}^{N_{\text{prod}}} \Big(r_o y_{o,i}^j + r_g y_{g,i}^j - (r_{wp} y_{wp,i}^j + r_{ep} y_{ep,i}^j) \Big) - \sum_{j=N_{\text{prod}}+1}^{N_w} \Big(r_{w_I} y_{w_I,i}^j + r_{e_I} y_{e_I,i}^j \Big) \Big] (1 + d_\tau)^{-\frac{t_i}{\tau}} \tag{3.4}$$

subject to

$$u_j^{\text{low}} \leq u_i^j \leq u_j^{\text{upp}}, \quad \forall i = 1, 2, ..., N_t,\ \forall j = 1, 2, ..., N_w \tag{3.5}$$

and

$$\sum_{j=1, N_{\text{prod}}+1}^{N_{\text{prod}}, N_w} u_i^j \leq C_{\text{total}}^v, \quad \forall i = 1, 2, ..., N_t, \quad v \in \{\text{inj}, \text{prod}\}, \tag{3.6}$$

where N_{prod} denote the number production wells respectively, J_i is the cumulative NPV over the $i-$th simulation time step, d_τ is the discount rate for a period of time τ, t_i is the sum total time from the start of production up to the $i-$th time step, $\Delta t = t_{i-1} - t_i$ is the difference between the time steps t_i and t_{i-1}. The scalars $r_o, r_g, r_{wp}, r_{w_I}, r_{ep}$, and r_{e_I} are the price of oil and gas, cost of water and EOR chemical/gas production and injection respectively. Further, $y^j_{o,i}, y^j_{g,i}, y^j_{wp,i}$, and $y^j_{ep,i}$ represent the total oil, gas, water, and EOR chemical/gas produced over the period of time Δt_i at the production well $j = 1, 2, ..., N_{\text{prod}}$ respectively. They depend on the reservoir states (more specifically, the states at the well-block). The quantities $y^j_{w_I,i}$ and $y^j_{e_I,i}$ denote the total water and EOR chemical/gas injected over the period of time Δt_i at the injection well $j = N_{\text{prod}} + 1, ..., N_w$ respectively and are components in the input vector $\mathbf{u}_i$.

In the constraints functions (3.5), the quantities u^{low}_j and u^{upp}_j are the lower and the upper bounds for the input variable u^j_i. This corresponds to the limitations on capacities of the respective injection and/ or production wells. In addition, the set of constraints (3.6) is defined on a specified input control type u^j_i (e.g., water rate) of a given set of injection wells, $j = 1, ..., N_{\text{prod}}$ and/ or production wells $j = N_{\text{prod}} + 1, ..., N_w$ for all time steps $i = 1, 2, ..., N_t$; C^v_{total} is a non-negative constant value specified based on the overall allowable capacity for the well type v, that's

$$C^v_{\text{total}} = \begin{cases} C^{\text{prod}}_{total}, & \text{if } j = 1, 2, 3..., N_{\text{prod}} \\ \\ C^{\text{inj}}_{total}, & \text{if } j = N_{\text{prod}} + 1, ..., N_w, \end{cases} \tag{3.7}$$

with "inj":=injection well and "prod":=production well. This type of constraint is applicable for cases where one considers grouping of input controls and as such, want the sum of control type values in each time step not to exceed a given limit.

For every given $\mathbf{U} \in \mathcal{D}$ satisfying underlying constraints, the evaluation of objective function (3.3) requires solving the corresponding reservoir model Equation (2.17) in other to obtain the components of output $\mathbf{Y}$ (see, Equations (2.21) and (3.2)). For this reason, the optimization problem (3.3) - (3.6) is called model-based.

Since $\boldsymbol{\theta}$ is fixed, without lost of generality, the objective function (3.3) shall be simply written as $J(\mathbf{U})$. It is a multivariate real-valued function

$$\begin{aligned} J : \mathbb{R}^{N_u} &\longrightarrow \mathbb{R} \\ \mathbf{U} &\longmapsto J(\mathbf{U}) \end{aligned} \tag{3.8}$$

that could be highly non-linear, because of the complexity of the reservoir model. For this reason, solving the constrained problem (3.3) - (3.6) analytically is not feasible. Instead, many studies (Brouwer and Jansen, 2002; Xu et al., 2018; Sarma et al., 2006; Chen and Oliver, 2010; Jansen, 2011a; Lorentzen et al., 2006) have provided different numerical solution methods to solve similar problems (e.g., water and polymer flooding optimization problems).

The solution methods are broadly grouped into two namely, gradient-based and non-gradient (or derivative free) based methods (Jesmani et al., 2020; Islam et al., 2020; Nocedal and Wright, 2006). Both solution techniques moves in the design solution space $\mathcal{D}$ in a special pattern to search for the optimal solution.

The gradient-based method utilizes the gradient information (computed analytically or by approximation) of objective function to make a search in the solution space while the derivative-free methods uses mainly the objective function values in a stochastic way at each optimization iteration. For production optimization, the derivative-free methods are proven to perform efficiently well for low dimensional problems, however become inefficient for high dimensional problems (Arouri and Sayyafzadeh, 2020). Different versions of solution methods following the derivative-free approach namely, genetic algorithm, particle swarm method etc., can be found in (Goldberg, 1989; Eberhart and Kennedy, 1995; Semnani et al., 2021; Chen et al., 2020)

Considering that the number of optimization variables of problem (3.3) - (3.6) is usually large and ranges between 100 and above, the gradient-based methods are more suitable for this problem. The present study proposes a solution technique for the constrained EOR problem that falls in the class of gradient-based methods.

4 Gradient-based optimization method

As mentioned in Section 3.1, the constrained EOR problem (3.3) - (3.6) is of high dimension and non-linear. To find its optimal solution using gradient-based methods requires the gradient information of the objective function at each optimization iteration. In this type of problem, access to analytic gradient is difficult. However, several interesting techniques for appropriately and efficiently approximating the gradient are available. It is noted that, poorly computed gradient schemes impact the number of iterations that a solution method takes to converge. Here, the two broadly used method of computing the approximate gradient of J, the adjoint and stochhastic methods, in reservoir optimization problems are discussed.

4.1 Adjoint method

The adjoint method (Jansen et al., 2008; Sarma et al., 2005) is known to be the most efficient and accurate way of computing the gradient of J with respective to the input vector $\mathbf{u}_i \in \mathbf{U}, \forall i = 1, 2, ..., N_t$. It is obtained by applying the necessary condition of optimality to the Lagrangian function arising from the objective function and its underlying dynamic state constraints (Stengel, 1994). For an arbitrary differentiable function f, the necessary optimality condition states that the gradient of f vanishes at its optimal point. It is from the classical theory of calculus (Nocedal and Wright, 2006; Snyman and Wilke, 2005).

From the formulation of J in (3.3), an infinitesimal change of J, denoted by δJ with respect to an infinitesimal change in component u_i^j of $\mathbf{u}_i$, denoted by δu_i^j is computed using product rule as:

$$\begin{aligned}\frac{\delta J}{\delta u_i^j} &= \frac{\partial J_i}{\partial \mathbf{u}_i}\frac{\partial \mathbf{u}_i}{\partial u_i^j} + \sum_{m=i}^{N_t}\Big(\frac{\partial J_m}{\partial \mathbf{y}_m}\frac{\partial \mathbf{y}_m}{\partial \mathbf{u}_i}\frac{\partial \mathbf{u}_i}{\partial u_i^j} + \frac{\partial \mathbf{y}_m}{\partial \mathbf{x}_m}\frac{\partial \mathbf{x}_m}{\partial \mathbf{u}_i}\frac{\partial \mathbf{u}_i}{\partial u_i^j}\Big)\\ &= \Big[\frac{\partial J_i}{\partial \mathbf{u}_i} + \sum_{m=i}^{N_t}\Big(\frac{\partial J_m}{\partial \mathbf{y}_m}\frac{\partial \mathbf{y}_m}{\partial \mathbf{u}_i} + \frac{\partial \mathbf{y}_m}{\partial \mathbf{x}_m}\frac{\partial \mathbf{x}_m}{\partial \mathbf{u}_i}\Big)\Big]\frac{\partial \mathbf{u}_i}{\partial u_i^j},\end{aligned} \tag{4.1}$$

for all $j = 1, 2, ..., N_w$ in the time steps $i = 1, 2, ..., N_t$.

In Equation (4.1), the calculation of the derivatives $\frac{\partial \mathbf{x}_i}{\partial \mathbf{u}_i}, i = 1, 2, ..., N_t$ require the solutions $\mathbf{x}_i$ to the corresponding non-linear algebraic equation (2.17) - (2.18). This dependence is taken into account by setting (2.17), (2.18), and (2.21) as additional constraints to the optimization problem (3.3) - (3.6). Here, the resulting problem is stated as follows (without the constraints (3.5) and (3.6) because they are handled separately):

$$\underset{\mathbf{U}\in\mathcal{D}}{\text{maximize}} \sum_{i=0}^{N_t-1} J_{i+1}(\mathbf{u}_{i+1}, \mathbf{y}_{i+1}) \tag{4.2}$$

subject to:

$$\mathbf{g}_{i+1}(\mathbf{u}_{i+1}, \mathbf{x}_{i+1}, \mathbf{x}_i, \boldsymbol{\theta}) = \mathbf{0}, \quad i = 0, 1, 2, ..., N_t - 1, \tag{4.3}$$

$$\mathbf{x}_0 - \bar{\mathbf{x}}_0 = \mathbf{0}, \tag{4.4}$$

and

$$\mathbf{y}_{i+1} - \mathbf{h}_{\boldsymbol{\theta},i+1}(\mathbf{u}_{i+1}, \mathbf{x}_{i+1}) = \mathbf{0}, \quad i = 0, 1, 2, ..., N_t - 1. \tag{4.5}$$

The Lagrangian function associated with (4.2) - (4.5) is given as:

$$L(\mathbf{U}, \mathbf{X}, \mathbf{Y}, \boldsymbol{\Lambda}, \boldsymbol{M}) = \sum_{i=0}^{N_t-1} \Big(J_{i+1}(\mathbf{u}_{i+1}, \mathbf{y}_{i+1}) + \boldsymbol{\lambda}_0^{\mathsf{T}}(\mathbf{x}_0 - \bar{\mathbf{x}}_0) + \boldsymbol{\lambda}_{i+1}^{\mathsf{T}} \mathbf{g}_{i+1}(\mathbf{u}_{i+1}, \mathbf{x}_{i+1}, \mathbf{x}_i, \boldsymbol{\theta}) + \boldsymbol{\mu}_{i+1}^{\mathsf{T}}(\mathbf{y}_{i+1} - \mathbf{h}_{\boldsymbol{\theta},i+1}(\mathbf{u}_{i+1}, \mathbf{x}_{i+1}))\Big), \tag{4.6}$$

where $\mathbf{X}$ retains all dynamic states $\mathbf{x}_{i+1}, \forall i = 0, 1, ..., N_t - 1$; $\boldsymbol{\Lambda} = \{\boldsymbol{\lambda}_i\}_{i=0}^{N_t-1}$ and $\boldsymbol{M} = \{\boldsymbol{\mu}_{i+1}\}_{i=0}^{N_t}$ are Lagrangian multipliers. On the application of necessary optimality condition to the variation of (4.6) with respect to the given variables (namely, components of $\mathbf{U}, \mathbf{X}, \boldsymbol{\Lambda}$, and $\boldsymbol{M}$ respectively), equations (A.7)-(A.14) are obtained.

Suppose (for the moment) that the input control $\mathbf{U}$ satisfies the constraints (3.5) - 3.6, then the system of differential equations (A.7) - (A.14) is called the adjoint model for the specific constrained EOR optimization problem (3.3) - (3.6).

Since Equations (A.7), (A.8), and (A.9) correspond to the prescribed initial state condition (2.18), the algebraic EOR model equation (2.17), and the output vector (2.21) respectively, they are instinctively satisfied.

Using the differential equation (A.10), the components of $\boldsymbol{M}$ are computed. Substituting $\boldsymbol{\mu}_{N_t} \in \boldsymbol{M}$ into Equation (A.11) gives $\boldsymbol{\lambda}_{N_t}$. Further, using the components of $\boldsymbol{M}$ and $\boldsymbol{\lambda}_{N_t}$, the remaining components of $\boldsymbol{\Lambda}$ are computed backwardly in time step $i = N_t - 1, N_t - 2, ..., 2, 1, 0$ using the differntial equation (A.12).

By substituting the respective Lagrangian multipliers into Equation (A.14), the gradient of objective function J with respect to the input control $\mathbf{u}_{i+1}, \forall i = 0, 1, ..., N_t - 1$ is computed as follows[1]:

$$\nabla_{\mathbf{u}_i+1} J := \frac{\partial L}{\partial \mathbf{u}_{i+1}} = \frac{\partial J_{i+1}}{\partial \mathbf{u}_{i+1}} + \boldsymbol{\lambda}_{i+1}^{\top} \frac{\partial \mathbf{g}_{i+1}}{\partial \mathbf{u}_{i+1}} + \boldsymbol{\mu}_{i+1}^{\top} \frac{\partial \mathbf{h}_{\boldsymbol{\theta},i+1}}{\partial \mathbf{u}_{i+1}}. \tag{4.7}$$

The adjoint procedures for computing the gradient of the objective function (3.3) is summarized in Appendix A.2 (see, Algorithm 3)

It is clear from the above demonstrations that the adjoint method of computing the gradient of J requires the information of the dynamic states $\mathbf{X}$ and the corresponding output $\mathbf{Y}$. Access to this information at all time steps is crucial. Although the adjoint method produces an accurate result, a specific implementation cannot be easily used if the model equations or selection of control variables change.

The non-reusable of adjoint equations is because, for instance, reservoir development with distinct recovery processes give rise to a different algebraic equations (2.17) and hence different reservoir simulators [2]. More so, computing the gradient of J by the adjoint method would require access to the reservoir simulator code to get information for the gradient (Sarma et al., 2005, 2006; Jansen, 2011a). Therefore, it is almost impossible to use the adjoint method while keeping the reservoir simulator as a black box.

To solve the problem of adjoint model reusability for well placement optimization problems, indirect methods were developed (see, e.g., Sarma and Chen (2008); Zandvliet et al. (2008)). However, computing the adjoint gradient without looking into the reservoir simulator is inevitable. This is one reason for exploring other, less intrusive, optimization methods

Further, the objective function (3.3) is highly multi-modal, i.e., it has multiple optimal solutions, see, e.g., Figure 4.1, because of its dependence on the reservoir simulator with several operational constraints. Consequently, the optimality condition used to compute the classical adjoint gradient is insufficient to guarantee an ascent direction leading to the best optimal solution. In this case, the optimization

[1]Equation (4.7) denotes the effect of varying the input control $\mathbf{u}_{i+1}$ on the objective function (3.3) while other variables are constant. At a non-optimal input control, $\nabla_{\mathbf{u}_i+1} J \neq \mathbf{0}$.. However, this non-zero valued vector is used as the gradient of J at the said control.

[2]The dynamic states and hence the output of the oil reservoir due to a given input control per time (see Section 2.1) are obtained by solving this algebraic equations

algorithm using the adjoint method is prone to finding a less desirable sub-optimal solution such as point A in Figure 4.1. There are other methods based on input control perturbations for computing the approximate gradient that can be efficient for high-dimensional and multi-modal problems.

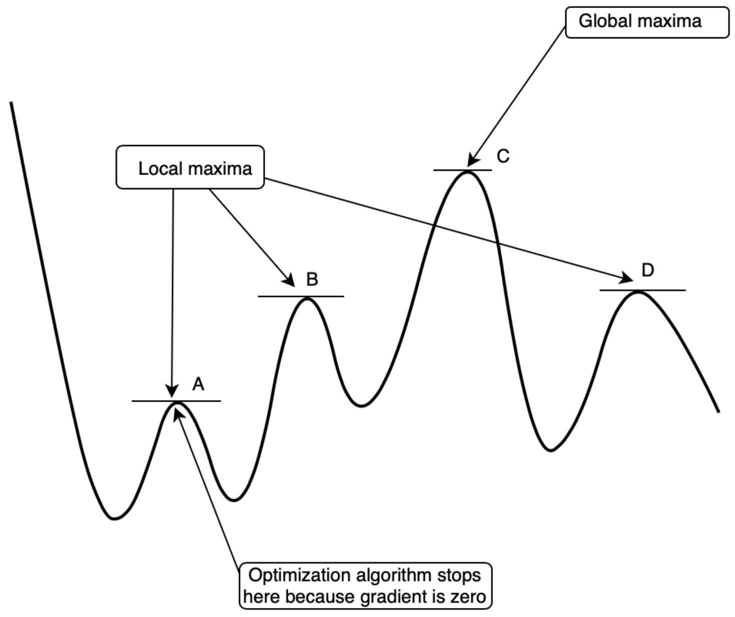


Figure 4.1: An example of one dimensional multi-modal function.

4.2 Stochastic method

The stochastic methods provide alternative and effective ways to approximate the gradient of J. Instead of using the adjoint model, stochastically perturbed input control with appropriate statistics are used to establish the gradient of J. The Finite Difference Stochastic Approximation (FDSA) is a classical stochastic gradient method that uses a finite difference scheme to approximate each component of the gradient function. It proves to give high accuracy and converges very fast for low dimension problems. For a problem with N_u control variables, the FDSA method perturbs each variable per time and uses $2N_u$ function evaluations to compute the gradient of J with respect to the variable. Hence in total, a full gradient requires $2N_u^2$ function evaluations which translates to $2N_u^2$ reservoir simulation runs. This process becomes computationally very expensive and inefficient to use for high dimension optimization problems (Jesmani et al., 2020; Zhou et al., 2013).

The efficiency of the FDSA method improves by simultaneously perturbing all the input variables using suitable statistics to compute the gradient function per time. This gives rise to the Simultaneous Perturbation Stochastic Approximatio (SPSA) technique. Here, a complete gradient computation requires only 2 function evaluations (equivalently, 2 simulation runs) (Zhou et al., 2013) instead of $2N_u^2$ evaluations by the FDSA method. The computational cost becomes greatly reduced using the SPSA technique.

The numerical experiment of Zhou et al. (2013) illustrates that the gradient function is estimated poorly by the SPSA method. It occurs where the contributions of different input variables to the overall change of the objective function considerably differ. Because of this, solution methods utilizing the SPSA gradients take a large number of iterations than the FDSA method before convergence. Therefore, a modified SPSA (called the SPSA-FDG) method was proposed, where a similar magnitude of contributions of different variables to the overall change of the objective function is systematically enforced. Other variants of the SPSA-FDG methods exist in the literature and there their theoretical connections are demonstrated in (Do and Reynolds, 2013).

Patelli and Pradlwarter (2010) introduced the Monte Carlo Gradient Approximation (MCGA) technique as alternatives to efficiently compute gradient of a generic function in high dimension settings. Here, N samples of input variables from a normal distribution and their respective function values are used to approximate the gradient using the Monte Carlo method. Xu et al. (2018) showed that the MCGA gradient is an unbiased estimation of the true gradient. Since it is Monte Carlo estimation, it is observed that the accuracy of the gradient increases as the number of perturbations tends to infinity.

Since the stochastic methods mentioned above use a given number of input variable realizations to compute the gradient function, they are often called the ensemble-based methods. The subsequent section discusses a version of the ensemble-based method utilized in this study.

4.2.1 Ensemble-based optimization method

The EnOpt approach is an efficient and non-intrusive stochastic method for approximating the gradient. Lorentzen et al. (2006) introduced the earlier version, and later developed into its current forms in (Chen et al., 2009; Chen and Oliver, 2010) and (Fonseca et al., 2014, 2017). For high-dimensional problems, the present state of the EnOpt method has been proven (using hypothesis testing) to give good quality gradient close to the adjoint method (Fonseca et al., 2015). For a given input vector $\mathbf{U}$, the EnOpt method uses an ensemble of input vectors normally distributed around $\mathbf{U}$ to estimate the gradient of the objective function at $\mathbf{U}$. The version of the EnOpt method presented here is similar to the Stochastic Simplex Approximate Gradient (StoSAG) technique (Fonseca et al., 2014).

For a single geology

With respect to a single geology $\boldsymbol{\theta}$, the present formulation of the EOR optimization problem (3.3) - (3.6) is called a deterministic problem. Here, one considers to solve the same non-linear algebraic equations (2.17) associated with the EOR method for different input controls to get their respective outputs using Equation (A.9). This information is then used to compute the objective function (3.3) values of the input vectors.

As mentioned in Section 3.1, for the deterministic case, $J(\mathbf{U}) := J(\mathbf{U}, \boldsymbol{\theta})$. Given an input vector $\mathbf{U}$, the EnOpt method estimates the gradient of J at $\mathbf{U}$ as follows: Sample N input vectors from a multivariate uniform distribution with mean $\mathbf{U}$ and a predefined covariance matrix $\mathbf{C_U}$ of size N_u. That's, $\mathbf{U}_k \sim \mathcal{N}(\mathbf{U}, \mathbf{C_U}), \quad \forall k = 1, 2, ..., N$. Otherwise, each perturbed vector $\mathbf{U}_k$ are obtained using the standard normal random vector $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbb{I})$ such that:

$$\mathbf{U}_k = \mathbf{U} + \mathbf{C}_{\mathbf{U}}^{1/2}\mathbf{z}_k, \quad \forall k = 1, 2, ..., N, \tag{4.8}$$

where $\mathbf{C}_{\mathbf{U}}^{1/2}$ is obtained by Cholesky decomposition (Higham, 2008; Jain, 2003) and $\mathbb{I}$ is an identity matrix size N_u. Although in this study the realizations of $\mathbf{U}$ follow a normal distribution, the same experiments can be performed with other types of distribution like in (Sarma and Chen, 2014).

With respect to each perturbed vector $\mathbf{U}_k$ in (4.8), the objective function $J(\mathbf{U}_k)$ is evaluated. This information is used to approximate the sample cross-covariance $\mathbf{C}_{\mathbf{U},J(\mathbf{U})}$ of the input control $\mathbf{U}$ and objective function $J(\mathbf{U})$ as follows:

$$\mathbf{C}_{\mathbf{U},J(\mathbf{U})} = \frac{1}{N-1}\Delta\mathbf{U}\Delta J^{\mathsf{T}}, \tag{4.9}$$

where

$$\Delta\mathbf{U} = [\mathbf{U}_1 - \mathbf{U}, \mathbf{U}_2 - \mathbf{U}, ..., \mathbf{U}_N - \mathbf{U}]$$

is a matrix of size $N_u \times N$ and

$$\Delta J = [J(\mathbf{U}_1) - J(\mathbf{U}), J(\mathbf{U}_2) - J(\mathbf{U}), ..., J(\mathbf{U}_N) - J(\mathbf{U})]$$

is a vector of size N.

Pre-multiplying the cross-covariance in (4.9) by the inverse of the sample covariance $\mathbf{C}_{\mathbf{U},\mathbf{U}}$ gives the approximate gradient of J at $\mathbf{U}$ by the EnOpt method. That's

$$\nabla_{\mathbf{U}} J \approx \mathbf{C}_{\mathbf{U},\mathbf{U}}^{-1} \mathbf{C}_{\mathbf{U},J(\mathbf{U})}. \tag{4.10}$$

Here,

$$C_{\mathbf{U},\mathbf{U}} = \frac{1}{N-1} \Delta\mathbf{U} \Delta\mathbf{U}^{\mathsf{T}}. \tag{4.11}$$

When $N < N_u$, the sample covariance $\mathbf{C}_{\mathbf{U},\mathbf{U}}$ becomes rank-deficient which could lead to a bang-bang optimal solution (i.e., solution with components switching between two extreme values over the entire simulation time steps). To avoid this situation, Chen et al. (2009) proposed a modified version of the gradient by pre-multiplying the right hand side of Equation (4.10) with $\mathbf{C}_{\mathbf{U},\mathbf{U}}^2$.

So far, the mathematical formulation of the gradient computation by the EnOpt method for the deterministic case has been presented. However, one main interest of the present study is to find optimal control strategies for EOR methods, taking into account the uncertainty description in reservoirs. Therefore, the subsequent section presents the the general EnOpt method for gradient computation in this case.

For multiple geologies

The process of reservoir modeling is highly uncertain because of limited knowledge about geological parameters such as porosity, etc. Yeten (2003) found that incorporating geological uncertainties into the water flooding optimization procedure reduces the uncertainty in optimization results. The numerical experiment of (Van Essen et al., 2009) showed that optimization over multiple geologies gives an optimal solution with increased economic value and reduced variance on application to the different geologies. This optimization framework is called robust optimization. On this account combined with the physical observation regarding possible impact of reservoir uncertainty on EOR effect in Section 1.4, it sounds more reliable to consider geological uncertainty when solving the optimization problem (3.3) - (3.6).

This study quantifies uncertain geological parameters in a given oil reservoir by choosing a suitable ensemble of geological realizations of the reservoir denoted by $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_m\}_{m=1}^{N_e}$. Here, N_e is the ensemble size. Also, assume that all geology are equally likely to be a true representation of the reservoir. The EOR optimization problem (3.3)-(3.6) is reconstructed as finding the optimal input control $\mathbf{U} \in \mathcal{D}$ that maximizes the expectation of J in (3.3) over $\boldsymbol{\Theta}$ denoted by $J_{\boldsymbol{\Theta}}(\mathbf{U})$ subject to the same set of constraints (3.5) - (3.6). Therefore, the new objective function is given by:

$$J_{\boldsymbol{\Theta}}(\mathbf{U}) = \frac{1}{N_e}\sum_{m=1}^{N_e} J(\mathbf{U}, \boldsymbol{\theta}_m) \approx \mathbb{E}_{\boldsymbol{\Theta}}[J(\mathbf{U})]. \tag{4.12}$$

The objective (4.12) with constraints (3.3)-(3.6) shall be referred to as the Robust EOR Optimization (REORO) problem in the rest of this study.

Different from cases with a single geology, one has to solve N_e-different algebraic equations (2.17) for a given input control resulting to N_e-outputs using (A.9). On this account, the objective function (4.12) value computes at the input control.

Several EnOpt formulations for computing the gradient of $J_{\boldsymbol{\Theta}}$ at a given input $\mathbf{U}$ exist in the literature see, e.g., Fonseca et al. (2017, 2014); Chen et al. (2009). In general, one samples $N \geq N_e$ input vectors using (4.8). Couple each perturbed vector $\mathbf{U}_k$ to N_e-geologies and compute $J(\mathbf{U}_k, \boldsymbol{\theta}_m)$, $\forall m = 1, 2, ..., N_e$. The average over all perturbations of (4.9) is computed, and the expectation of the gradient of $J_{\boldsymbol{\Theta}}$ follows from Equation (4.10).

In a more mathematical sense, assume that one seeks to find the approximate gradient of $J_{\boldsymbol{\Theta}}$ (equivalently, its expected value) at an input control $\mathbf{U}^l$ and suppose also that different $N-$samplings of $\mathbf{U}^l$ are drawn using (4.8) for each geology $\boldsymbol{\theta}_m$ and denote by $\{\mathbf{U}_{m,k}^l\}_{k=1}^N, \forall m = 1, 2, ..., N_e$. For a given $\boldsymbol{\theta}_m$, the Taylor's expansion of J about $\mathbf{U}^l$ gives:

$$J(\mathbf{U}, \boldsymbol{\theta}_m) - J(\mathbf{U}^l, \boldsymbol{\theta}_m) = \nabla_{\mathbf{U}} J(\mathbf{U}^l, \boldsymbol{\theta}_m)^{\mathsf{T}}(\mathbf{U} - \mathbf{U}^l) + O(||\mathbf{U} - \mathbf{U}^l||^2), \tag{4.13}$$

Pre-multiply both sides of (4.13) by $(\mathbf{U} - \mathbf{U}^l)$ and set $\mathbf{U} := \mathbf{U}^l_{m,k}$ lead to:

$$[J(\mathbf{U}^l_{m,k}, \boldsymbol{\theta}_m) - J(\mathbf{U}^l, \boldsymbol{\theta}_m)](\mathbf{U}^l_{m,k} - \mathbf{U}^l) = \nabla_{\mathbf{U}} J(\mathbf{U}^l, \boldsymbol{\theta}_m)^{\mathsf{T}}(\mathbf{U}^l_{m,k} - \mathbf{U}^l)(\mathbf{U}^l_{m,k} - \mathbf{U}^l) + O(||\mathbf{U}^l_{m,k} - \mathbf{U}^l||^3). \tag{4.14}$$

Here $O(||\mathbf{U}^l_{m,k} - \mathbf{U}^l||^3)$ is the remaining terms containing higher order (≥ 3) of $(\mathbf{U}^l_{m,k} - \mathbf{U}^l)$. This study assumes the magnitude of $(\mathbf{U}^l_{m,k} - \mathbf{U}^l)$ to be very small (tends to zero). As such, terms with higher order (≥ 3) of this difference become approximately negligible. Therefore, (4.14) leads to

$$[J(\mathbf{U}^l_{m,k}, \boldsymbol{\theta}_m) - J(\mathbf{U}^l, \boldsymbol{\theta}_m)](\mathbf{U}^l_{m,k} - \mathbf{U}^l) \approx \nabla_{\mathbf{U}} J(\mathbf{U}^l, \boldsymbol{\theta}_m)^{\mathsf{T}}(\mathbf{U}^l_{m,k} - \mathbf{U}^l)(\mathbf{U}^l_{m,k} - \mathbf{U}^l). \tag{4.15}$$

Since (4.15) is approximately true for all $m = 1, 2, ..., N_e$ and $k = 1, 2, ..., N$, then it is not hard to see that the following expression holds:

$$\frac{1}{N}\sum_{k=1}^{N}\frac{1}{N_e}\sum_{m=1}^{N_e}[J(\mathbf{U}^l_{m,k}, \boldsymbol{\theta}_m) - J(\mathbf{U}^l, \boldsymbol{\theta}_m)](\mathbf{U}^l_{m,k} - \mathbf{U}^l) \approx \frac{1}{N}\sum_{k=1}^{N}\frac{1}{N_e}\sum_{m=1}^{N_e}\nabla_{\mathbf{U}} J(\mathbf{U}^l, \boldsymbol{\theta}_m)^{\mathsf{T}}(\mathbf{U}^l_{m,k} - \mathbf{U}^l)(\mathbf{U}^l_{m,k} - \mathbf{U}^l). \tag{4.16}$$

Taking the expectations, first with respect to $\mathbf{U}^l$ and secondly with respect to $\boldsymbol{\Theta}$ of the right hand side expression in (4.16), and then substituting (4.12) and (4.11) gives the following:

$$\mathbb{E}_{\mathbf{U}^l}\Big[\mathbb{E}_{\boldsymbol{\Theta}}\Big(\frac{1}{N}\sum_{k=1}^{N}\frac{1}{N_e}\sum_{m=1}^{N_e}\nabla_{\mathbf{U}} J(\mathbf{U}^l, \boldsymbol{\theta}_m)^{\mathsf{T}}(\mathbf{U}^l_{m,k} - \mathbf{U}^l)(\mathbf{U}^l_{m,k} - \mathbf{U}^l)\Big)\Big] = \frac{1}{N}\sum_{k=1}^{N}\mathbb{E}_{\mathbf{U}^l}\Big[(\mathbf{U}^l_{m,k} - \mathbf{U}^l)(\mathbf{U}^l_{m,k} - \mathbf{U}^l)\Big]\nabla_{\mathbf{U}}\Big(\frac{1}{N_e}\sum_{m=1}^{N_e}\mathbb{E}_{\boldsymbol{\Theta}}(J(\mathbf{U}^l, \boldsymbol{\theta}_m))\Big) = C_{\mathbf{U}^l,\mathbf{U}^l}\nabla_{\mathbf{U}} J_{\boldsymbol{\Theta}}(\mathbf{U}^l). \tag{4.17}$$

This implies that the left hand side expression in (4.16) is an unbiased estimator of the preconditioned gradient function obtained in (4.17). Consequently, the desired approximate gradient is arrived at using (4.17) and (4.16). That's:

$$\nabla_{\mathbf{U}} J_{\boldsymbol{\Theta}}(\mathbf{U}^l) \approx C^{-1}_{\mathbf{U}^l,\mathbf{U}^l}\Big[\frac{1}{N}\sum_{k=1}^{N}\frac{1}{N_e}\sum_{m=1}^{N_e}[J(\mathbf{U}^l_{m,k}, \boldsymbol{\theta}_m) - J(\mathbf{U}^l, \boldsymbol{\theta}_m)](\mathbf{U}^l_{m,k} - \mathbf{U}^l)\Big]. \tag{4.18}$$

From (4.18), it is seen that the gradient requires $N \times N_e-$function evaluations (equivalently, solving $N \times N_e-$algebraic equations (2.17)). Although, this is computationally very expensive, the gradient quality increases with increase in N as demonstrated by Fonseca et al. (2015) using hypothesis testing for the Rosenbrock's problem with uncertainty, see Figure 4.2.

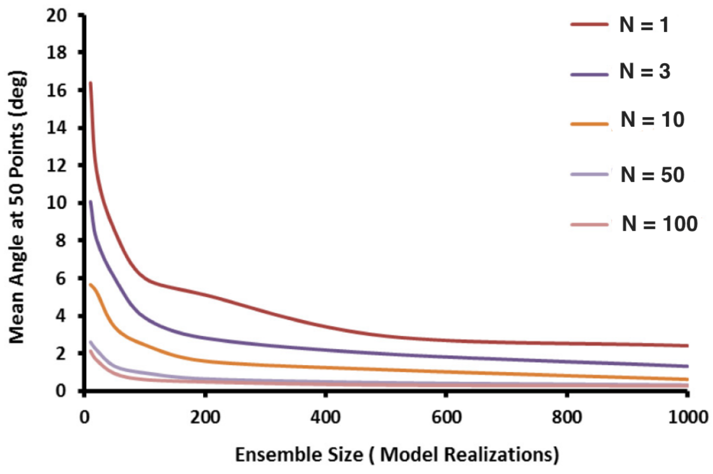


Figure 4.2: Variation of gradient quality (decrease in mean angle implies increase in quality) with the number of perturbed input controls for each model realization (Fonseca et al., 2015) .

Moreover, Fonseca et al. (2015) demonstrates using a practical example that with $N = 1$ in (4.18) results in an improved optimal solution compared to the standard EnOpt approach[3], and thus present study follows this approach. Here, a one-to-one coupling of the perturbed vectors with geologies is incorporated. The EnOpt procedures for calculating the gradient of J_{Θ} is further summarized in Algorithm (1).

4.3 Optimization algorithm

Given the procedures for computing the gradient of the objective function (4.12) using the EnOpt method, this section presents the optimization algorithm utilized

[3]The standard EnOpt method presented by Chen et al. (2009) computes (4.18) by setting $\mathbf{U}^l := \frac{1}{N_e}\sum_{m=1}^{N_e}\mathbf{U}^l_{m,k}$ and $J(\mathbf{U}^l, \boldsymbol{\theta}_m) := \frac{1}{N_e}\sum_{m=1}^{N_e} J(\mathbf{U}^l_{m,k}, \boldsymbol{\theta}_m)$ with $N = 1$.

Algorithm 1: Computation of gradient of the objective function (4.12) using the robust EnOpt method in a general setting.

```
input: Prescribe input control $\mathbf{U} = \{\mathbf{u}_{i+1}\}_{i=0}^{N_t-1}$, sample size $N$, ensemble size
       $N_e$, and covariance matrix $C_{\mathbf{U}}$.
output: $\nabla_{\mathbf{U}} J_{\mathbf{\Theta}}$.
1  begin
2      forward simulation with OPM-Flow
3          Compute output vector $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1}^{N_t}$ using Equation (2.21) for each
             $\boldsymbol{\theta}_m$, to evaluate $J(\mathbf{U}, \boldsymbol{\theta}_m)$ $m = 1, 2, ..., N_e$ using (3.3).
4      simultaneous perturbation
5          Using (4.8), generate a sample space of size $N \times N_e$ with mean $\mathbf{U}$
             and covariance matrix $C_{\mathbf{U}}$.
6          forward simulation with OPM-Flow
7              for $m = 1 = 1, 2, ..., N_e$ do
8                  Couple different $N$ perturbed vectors with $\boldsymbol{\theta}_m$ until the sample
                     space is exhausted
9                  for $k = 1, 2, ..., N$ do
10                     Compute output vector $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1}^{N_t}$ using Equation (2.21)
                         for $\mathbf{U}_{k,m}$ and $\boldsymbol{\theta}_m$ to evaluate $J(\mathbf{U}_{k,m}, \boldsymbol{\theta}_m)$ using (3.3)
11     Compute $\nabla_{\mathbf{U}} J_{\mathbf{\Theta}}$ using Equation (4.18).
```

in this study to arrive at the desired optimum of the robust EOR optimization problem. Although it is difficult to find the global solution to the problem, it is possible to arrive at a desirable approximate solution to the global optimum using a suitable direct-search approach. Here, focus is on the minimization equivalence of the REORO problem since $-\min(-J_{\mathbf{\Theta}}) = \max J_{\mathbf{\Theta}}$.

In this study, the line search technique is used to find the approximate solution (Nocedal and Wright, 2006; Snyman and Wilke, 2005). In general, the line search algorithm starts with a user-defined initial estimate $\mathbf{U}^0$ to the optimum $\mathbf{U}^*$ and generates a sequence of improved (in the sense of enough decrease in function value) estimates or iterates $\mathbf{U}^0, \mathbf{U}^1, \mathbf{U}^2, \mathbf{U}^3, ...$, by successively searching from each estimate along a descent direction to obtain the subsequent estimate. The process continues until a given convergence criterion has been sufficiently satisfied. Hence, the last iterate becomes the approximate optimum solution.

The important parts of any line search method are the search direction denoted by $\mathbf{d}_l$ and the step length β^l to take along $\mathbf{d}^l$ at each $l-$th optimization iteration to get an improved estimate. Algorithm 2 states the general exact line search procedures to find the optimum point of J_Θ in (4.12). It is exact because the step length is computed by solving exactly a one-dimensional minimization sub-problem at each iteration, see, Line 5 of Algorithm 2. Further, Figure 4.3 illustrates the structure of Algorithm 2 for an arbitrary function.

Algorithm 2: General procedure of an exact line search method.

```
input: Prescribe input control $\mathbf{U}^0$, maximum number of iterations $\mathbb{N}_n$, and
       positive tolerance $\epsilon_1$
output: optimum $\mathbf{U}^*$
1 begin
2   for $l \in \mathbb{N}_n$ do
3     choose $\mathbf{d}^l$
4     while $|J_\Theta(U^{l+1}) - J_\Theta(U^l)| > \epsilon_1$ do
5       find $\beta^l$ : $\min J_\Theta(\mathbf{U}^l + \beta^l \mathbf{d}^l)$
6       choose $\mathbf{U}^{l+1} := \mathbf{U}^l + \beta^l \mathbf{d}^l$
7       set $l := l + 1$
8   return $\mathbf{U}^* := \mathbf{U}^{l+1}$
```

This study considers a preconditioned gradient descent inexact line search method. Here, the search direction is computed by preconditioning the approximate gradient formulation in (4.18) with input control covariance. That's

$$\mathbf{d}^l = C_{\mathbf{U}^l,\mathbf{U}^l} \nabla_{\mathbf{U}^l} J_\Theta(\mathbf{U}^l), \quad \forall l \in \mathbb{N}_n. \tag{4.19}$$

Chen et al. (2009) and Fonseca et al. (2017) showed that using a preconditioned gradient as the search direction gives a smoother direction. The gradient preconditioned with the square of covariance matrix $C_{\mathbf{U}^l,\mathbf{U}^l}$ is called the regularized or smoothed approximate gradient.

Since the solution search space $\mathcal{D}$ of the REORO problem has infinitely many points, it is therefore impossible to find a fixed global optimal covariance matrix that can generate ensemble of appropriate controls for gradient computation in all optimization iterations. For this reason, we use the Covariance Matrix Adaptation

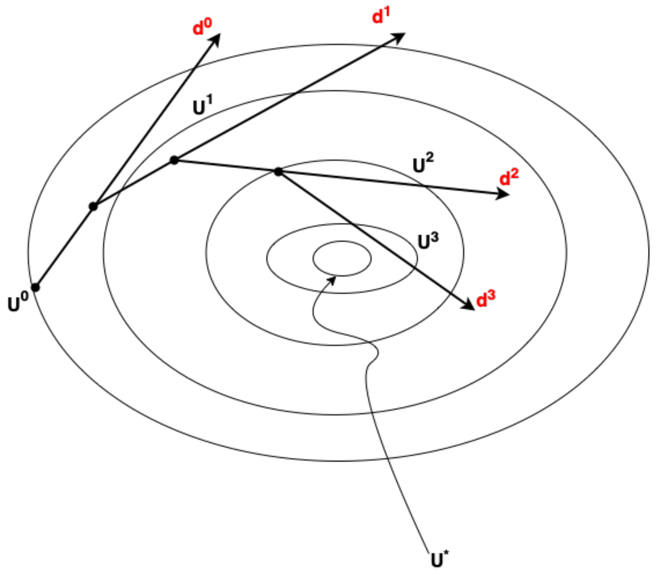


Figure 4.3: Contour plot of a function with sequence of estimates from line search method approaching the optimum point $\mathbf{U}^*$.

(CMA) technique to continuously improve the distribution covariance matrix for good quality gradient, see **PAPER III**.

Many accurate one-dimensional minimization methods (for Line 5 in Algorithm 3) exist in the literature, e.g., quadratic and cubic interpolation techniques of Powell (1964); Davidon (1959) for problems with smooth functions, bracketing (or Fibonacci search) method of Kiefer (1957) is more suited for poorly conditioned function. However, the computational cost associated with solving exactly the minimization problem of Line 5 in Algorithm 2 is demanding considering the size and complexity of the objective function (4.12). To reduce cost, this study considers an inexact approach similar to the backtracking technique (Nocedal and Wright, 2006) and can be found in **Paper IV**.

4.4 Constraint handling technique

At each stage of constrained optimization, one must ensure the input control variables honor the underlying constraints. For the REORO problem in this study, the bound constraints (3.5) is taking care of using an appropriate bijective linear function which administers suitable optimization domain to the original search space of the input control **U**, see, **PAPER I**. In the administered space, we carry out optimization procedures, and to evaluate the function value of a given control, its equivalence in the original solution space is used. The equivalence is obtained by using the inverse of the linear transformation.

In general, the constraints (3.6) are usually handled by the reservoir simulator. Setting an upper limit for a group of a given control type, e.g., water rate at injection wells, is managed by the group operational constraints imposed by the simulator. In situations where these constraints are violated, the simulator switch from rate to pressure control.

PAPER I and **PAPER III** presents optimization results obtain by using a special case ($N = 1$) of the solution method presented in Algorithm 2 to solve the REORO problems associated with synthetic reservoir models.

4.5 Application for robust EOR quantification and ranking

The evaluation of the economic benefit of an EOR strategy for a given reservoir field is in this study called the EOR value quantification. In **PAPER III**, we carry out the value quantification of EOR methods with traditional water flooding as a reference point. In this, we weigh the performance of the optimal control strategy (obtained by solving the associated REORO problem as presented in Section 4.3) for each EOR method over that of water flooding in terms of fluid productions and NPV value. These results are, in turn, used to rank the EOR methods for the examples considered.

For the quantification and ranking of EOR strategies, this study considers three different oil reservoir models, namely 2D five-spot, 3D Reek, and 3D Olympus, with distinct rock fluid properties. The Reek and the Olympus models have been developed to mimic the North Sea Reservoirs.

The five-spot field

This is an in-house 2D reservoir model with three-phase flow (including oil, water, and gas). The model has one injection and four production wells spatially arranged in a five-spot pattern as shown in Figure 4.4. The field is uniformly discretized into 50×50 grid cells, with $\Delta x = \Delta y = 100$ m. On average, the reservoir has approximately 30% porosity with heterogeneous permeability distribution. The initial reservoir pressure is 200 bar. The initial average oil and water saturations are respectively 0.6546 and 0.3454. The original oil in place (OOIP) is 4.983×10^6 sm^3. Fluid properties are similar to that of a light oil reservoir. The viscosity (in cP) for saturated oil at varying bubble point pressure lies in the interval $[0.1, 0.56]$ and viscosity of water is 0.2 cP. The densities of oil and water are taken as 732 kg/m^3 and 1000 kg/m^3 respectively.

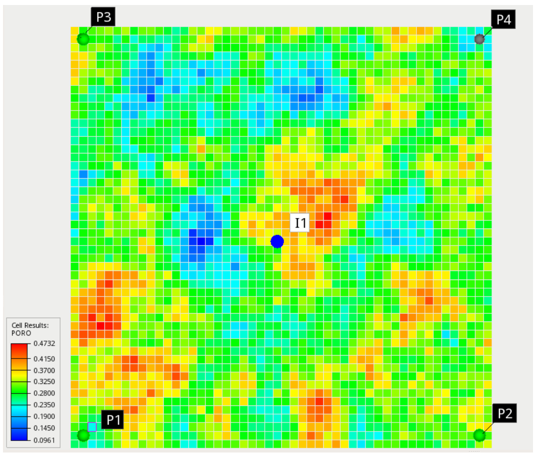


Figure 4.4: Porosity distribution of the five-spot field.

The Reek field

This is a 3D reservoir model designed by Equinor. It has three phases (including oil, water, and gas). The reservoir model is defined on an irregular grid system of dimensions $40 \times 64 \times 14$. This results in a total of 35840 grid cells with different sizes. There are three zones (UpperReek, MidReek, and LowerReek) with six faults and varying porosity and permeability. The model is equipped with five production and three injection wells (see Figure 4.5). All the three injectors are positioned in the water saturated zones, while the producers are spatially distributed throughout the oil containing region based on engineering intuition. The oil viscosity (in cP) is modeled as a function of bubble point pressure and lies in the closed interval $[0.09, 1]$. The water viscosity is 0.25 cP. However, for this reservoir, uncertain properties such as facies, porosity, permeability, oil-water contacts, relative permeability, and transmissibility across five faults (out of the six) are quantified using 50 geological realizations. Although there are 35840 grid cells, not all of them are active. The number of active cells varies with geology. On average, the original oil in place (OOIP) is 4.831×10^7 sm^3. The initial average oil, water, and gas saturations are 0.1436, 0.8509, and 0.0055 respectively. The reservoir model is divided into two saturation regions. Figure 4.6 shows the relative permeability of the 50 geo-models in the saturation regions.

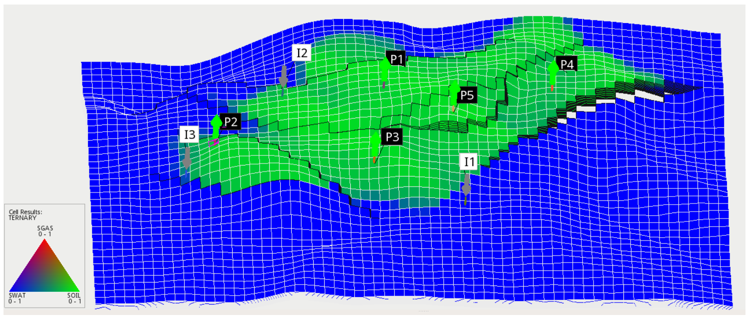
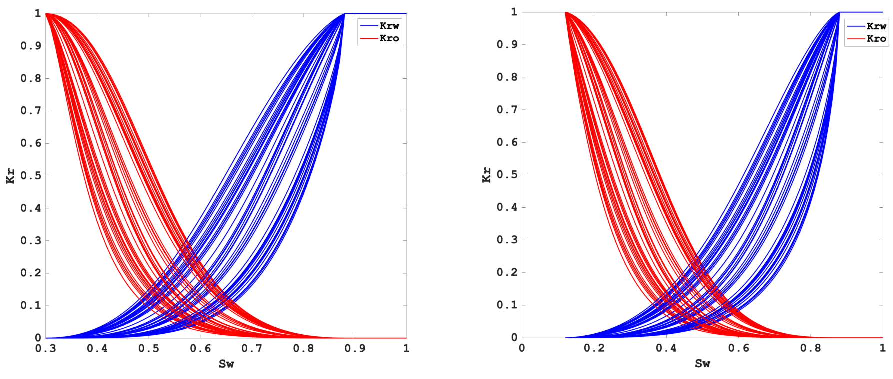


Figure 4.5: The initial saturation map (for oil, water and gas) for a geo-model of the Reek field.



(a) Pressure (b) Saturation

Figure 4.6: Relative permeability of the 50 geo-models of the Reek field. (a) Relative permeability in the saturation region 1. (b) Relative permeability in the saturation region 2.

The Olympus field

The Olympus field is a 3D synthetic reservoir model (see Figure 4.7) with two phases (including oil and water). It is designed by the Netherlands organization for applied Scientific Research (TNO). A detailed description of the reservoir model is found in (Fonseca et al., 2018). The dimension of the model is $118 \times 181 \times 16$.

In total, there are 341728 grid cells of which 192750 are active. The size of each grid cell is 50m $\times$ 50m $\times$ 3m. The model has 16 layers, which are separated into two zones (top and bottom zones) by an impermeable shale. Four different facies types, namely the channel sand, shale, coarse and fine sands, are modeled in the different layers. The porosity and permeability distributions for each facies in these layers are generated using a geo-statistical model. There are 18 wells in the Olympus field, namely 11 producers and 7 injectors. We consider the same fluid properties and their respective distributions presented in Fonseca et al. (2018) except that we have slightly increased the oil viscosity values. Now, the range of values (in cP) for the oil viscosity modeled as a function of bubble point pressure is taken as [20.5, 40.5]. The increase in viscosity is to ensure heavier oil in the reservoir. The uncertain properties in the reservoir are the facies, porosity, permeability, net-to-gross ratio, initial water saturation and transmissibility across the faults, and are quantified using 50 geological realizations. On average, the original oil in place (OOIP) is 4.95×10^7 sm^3. The oil and water saturation vary from one model to another. The Olympus field has four saturation regions. The relative permeability curves in the four regions are depicted by Figure 4.8. Unlike in the Reek field, we used the same saturation region for all the 50 geological realizations.

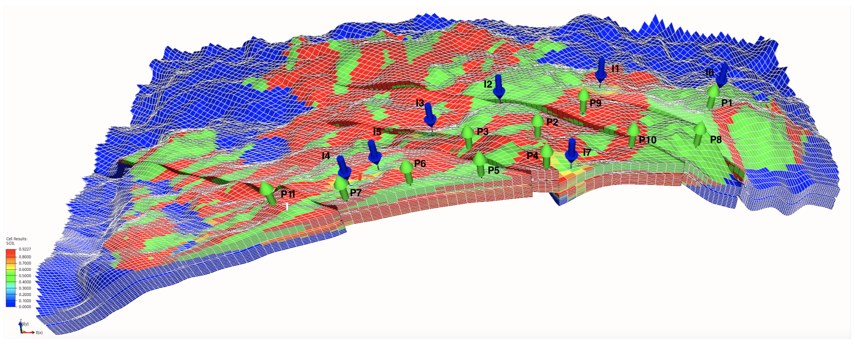
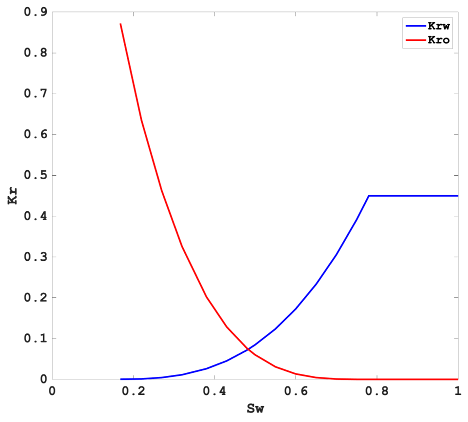


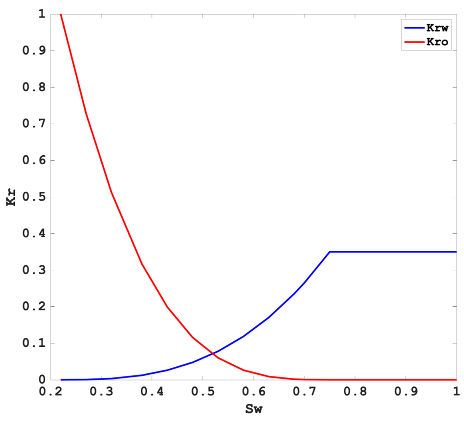
Figure 4.7: The initial oil saturation map for a geological realization of the Olympus field.

4.6 Improved EnOpt method for constrained optimization

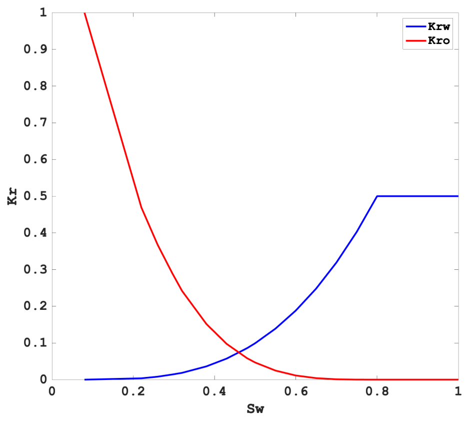
The constraint handling procedure presented in Section 4.4 are common practices mostly used in reservoir optimization problems, see e.g., Li and Reynolds (2011); Do and Reynolds (2013); Xu et al. (2018); Fonseca et al. (2014). The procedure



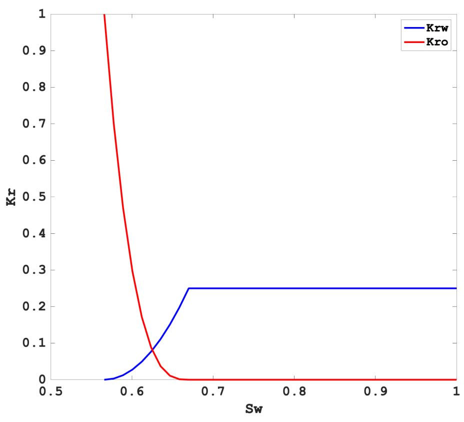
(a) Region 1



(b) Region 2



(c) Region 3



(d) Region 4

Figure 4.8: Relative permeability in the four saturation regions of the Olympus field.

works effectively for simple linear constraints like (3.5), especially in cases where there is a large number of them. Aside from the gradient quality being affected, as demonstrated in **PAPER II**, it is difficult to apply the procedure for complicated linear or non-linear constraints.

When the gradient quality is poor, this can translate to the need for a large number of iterations before the optimization algorithm converges to the desired optimum, as is the case for the 2D analytical Rosenbrock's problem considered in

PAPER II. For practical reservoir problems, poor gradient quality resulting from inappropriate constraint handling techniques will not only cause a large number of iterations but also lead to a less desirable sub-optimal solution.

The Lagrangian method is by far the most accurate technique for handling any constraint (Nocedal and Wright, 2006). Here, the constrained problem is transformed to an unconstrained subproblem by the introduction of the so-called Lagrange multiplier. The disadvantage of this method is that it requires constraint gradient with respective to the input variables and, consequently, adds to the computational cost of the overall optimization algorithm utilizing it. Zhang et al. (2016) and Lu et al. (2017) used the augmented Lagrangian method to solve the constrained optimization problem associated with water flooding. Their works provide an approximate procedures to estimate the Lagrange multipliers.

Our propose methodology in **PAPER II** utilizes the exterior penalty(EP) method to handle constraints. Given a constrained minimization problem involving a linear objective function $f : \mathbb{R}^{N_u} \longrightarrow \mathbb{R}, f(\mathbf{u}) = \zeta u_1, \zeta > 0$ as shown in Figure 4.9 and an inequality constraint $u_1 - \xi \geq 0$ with $\xi > 0$. In general, the EP method converts the constrained problem to a sequence of unconstrained sub-problems where a new objective function called the penalty function p_f is defined (Rao, 2019).

The p_f is constructed by adding an EP term, which depends linearly on an increasing sequence of penalty parameters $\{r_k\}_{k=1}^{\infty}$ (for constraint violation) to f. The penalty term is chosen such that its value will tend to zero at infeasible points approaching constraint boundaries and tends to infinity as infeasible points moves away from the constraint boundaries. Consequently, the value of p_f also tends to infinity as the infeasible points moves away from the constraint boundaries. This behaviour can be seen in Figure 4.9. On the application of unconstrained minimization algorithm (see Section 4.3) to each p_f as r_k tends to infinity, the subsequent optimum of p_f is found to approach (see Figure 4.9) and eventually coincides the optimum of f.

The EP method is found to be superior to the traditional constrained handling and outperformed the classical augumented Lagrangian method for the analytical and water flooding examples considered in **Paper II**.

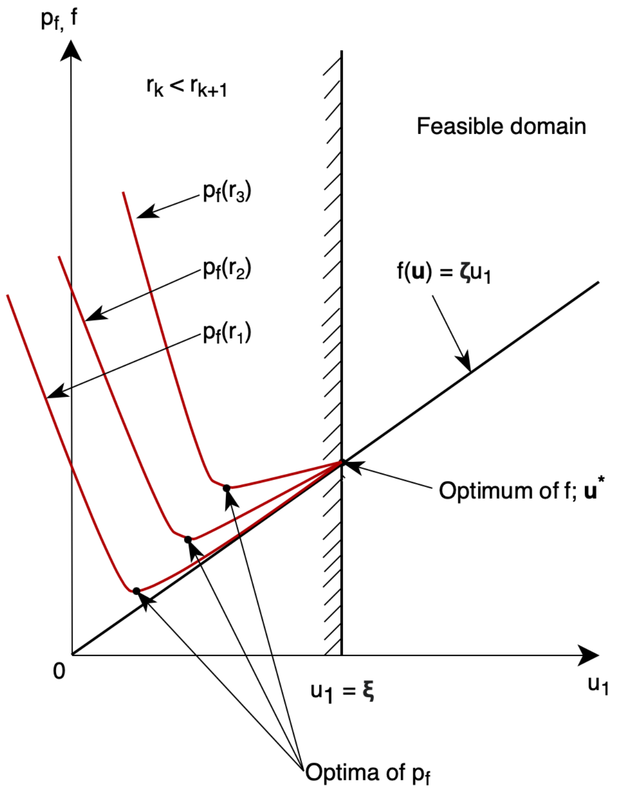


Figure 4.9: Exterior penalty method for a linear function.

5 Model order reduction for EOR optimization

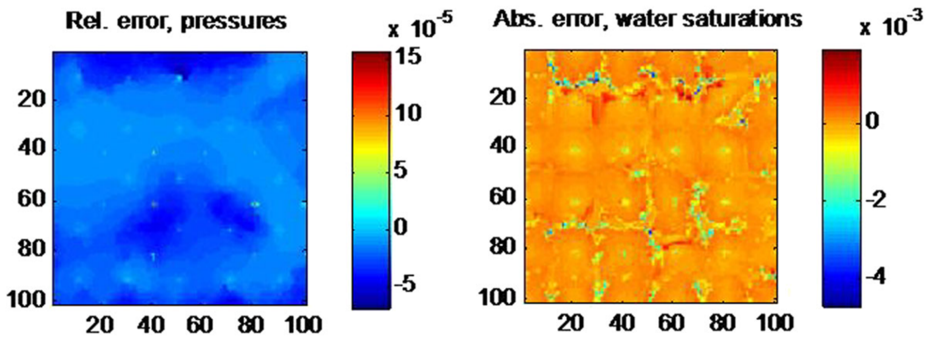
Previously, the solution method for optimizing the robust reservoir NPV function (4.12) under the EOR strategy is presented. Also, this can be called model-based PDE-constrained optimization since evaluating the objective function (or NPV) at a given input control relies on reservoir simulation, which involves solving some system of PDEs describing fluid flow in the reservoir of interest. As described earlier, the solution to PDEs gives the time evolution of the reservoir state, and hence its response, as a result of the input control. That's, to every EOR injection strategy, the reservoir response in terms of fluid production is measured. On this account, we compute the NPV. When a reservoir model contains large number of grid blocks or complicated underlying physics, the computation of the reservoir simulation, and hence the objective function relying on it, becomes demanding. Since a good quality gradient approximation requires several function evaluations, the EnOpt method becomes computational inefficient for large PDE-constrained optimization problems with complicated reservoirs. The proposed approach of this study on how to deal with the computationally expensive objective function for an efficient optimization algorithm is discussed.

5.1 Intrusive ROM method for reservoir simulation

Model-based optimization requires multiple reservoir simulations, especially when computing the approximate gradient using the EnOpt formulation (4.18), which requires several function evaluations. The costly computation of reservoir simulation has motivated the development of ROM techniques for accurate approximate of the reservoir simulation.

Given a problem of solving a coupled system of PDEs such as in reservoir simulations, which involves a large number of degrees of freedom, the ROM techniques fall in the category of methodologies for building an alternative representation of the problem with significantly fewer degrees of freedom. In other words, they simplify high-fidelity, complex models by projecting their required behaviors onto less complicated models.

The ideal goal of ROM techniques is to speed up computation without compromising too much solution accuracy; see Figure 5.1. In this experiment, Durlofsky (2010) built a ROM for a two-dimensional reservoir model to predict water flooding effect on reservoir states. With the ROM, they were able to speed up computation by a factor of 3.



(a) Pressure (b) Saturation

Figure 5.1: Relative differences in pressure (left) and absolute differences in saturation (right) between full order and reduced order model simulations for the same water flooding input control (Durlofsky, 2010).

Most ROM procedures developed for reservoir simulation utilize POD. The idea of POD originates from the domain of fluid dynamics where it is used to obtain low dimensional approximation of turbulent fluid flows (Holmes et al., 2012). In general, POD simply is the decomposition of a physical field (e.g., pressure and saturation in reservoir modeling), based on the different variables influencing its behavior into a discrete set of deterministic orthonormal basis functions. In another term, POD generates low-order models using snapshots from a forward simulation with the original high-order model (Van Doren et al., 2006).

In reservoir simulation, POD-based ROM techniques involve one or more high-fidelity training simulations with the original model at designated input controls, saving snapshots (reservoir states) at a number of time steps from these simulations, and then constructing a set of basis functions from these snapshots (the high-fidelity simulation with the original model shall be called the full order model (FOM) simulation). The basis functions are obtained from a singular value decomposition (SVD) of the snapshot matrix. In this case, efficiency is attained because only relatively small basis functions are retained and with linear combination can represent the subsequent reservoir flow solutions or states (Cardoso and Durlofsky, 2010).

Van Doren et al. (2006); Heijn et al. (2004); Kaleta et al. (2011); Astrid et al. (2011) used the above mentioned POD based ROM procedures for different reservoir flow applications and demonstrated a quite modest speedup in computations. It is observed that as non-linearity in the reservoir problem increases, the complexity in terms of number of basis functions required for appropriate approximation by the ROM techniques increases and thus impact ROM efficiency. For instance, see Cardoso et al. (2009), where the problem involves strong gravitational effect and highly nonlinear relative permeability maps. The increase in complexity with non-linearity makes the standard POD ROM technique less robust and attractive for a more evolving practical applications.

The limitations associated with the POD technique is relaxed using some linearization procedures, the trajectory piecewise linearization (TPWL), first introduced by Rewienski and White (2003). Here, the POD representation is combined with the linearization of the discrete-time flow equations (2.17) and the Jacobian around some saved states, which are generated using the FOM model in one or more training runs. Moreover, the linearized representation is then projected onto a low-order subspace. Major contribution of this approach is noticed in the computation speedup because POD is performed with the linearized equations rather than with the nonlinear equations.

The POD and POD-TPWL based ROM techniques highlighted above have been successfully utilized with gradient-based optimization algorithms to solve water flooding problems, see e.g., Van Doren et al. (2006); Durlofsky (2010). In these cases, the reduced model replaces the FOM model to simulate the reservoir responses to different input controls for their objective function evaluations. For example, Figure 5.2 compares the results from different versions of ROM-based optimization algorithms with that of FOM-based algorithm. However, the problem with developing ROM using these techniques is that they are highly intrusive because of the need to access the underlying flow equations. Moreover, efficiency of intrusive ROM methods for geologically heterogeneous reservoirs with varying structures and under EOR conditions still remain unclear.

5.2 Non-Intrusive ROM method for reservoir simulation

Many non-intrusive reduced order modeling (NIROM) methods have emerged and recently utilized for reservoir flow applications; see e.g., Xiao et al. (2016) for a complete list of references. Here, the term “non-intrusive" implies that the

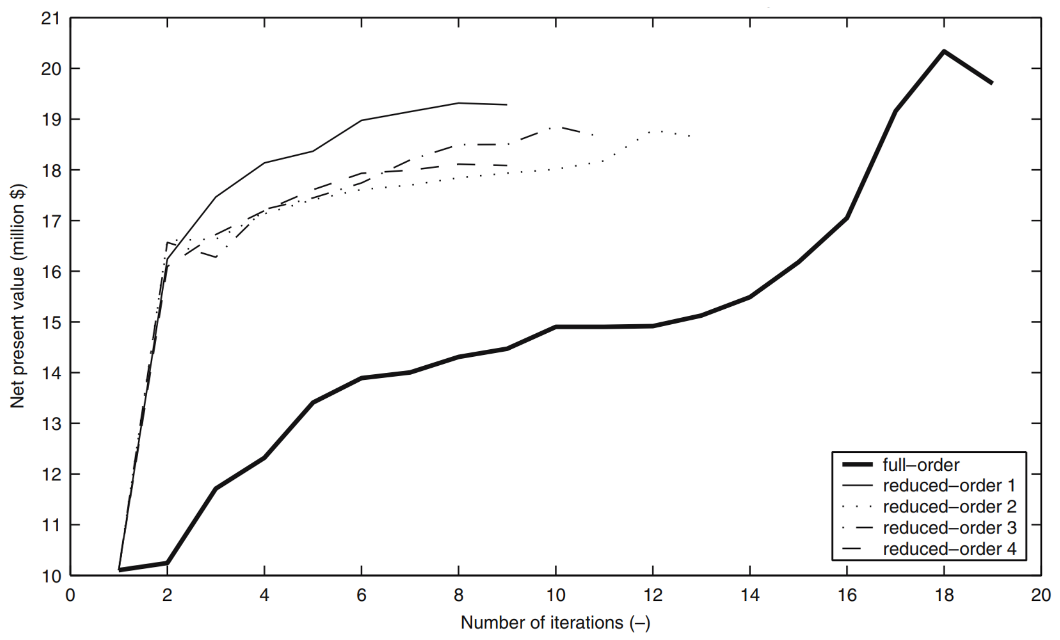


Figure 5.2: Comparisons of NPV versus number of iterations for different ROM-based and FOM-based optimization algorithms for water flooding optimization problem (Van Doren et al., 2006).

reservoir simulator remains as a black-box with the focus to approximate it.

Over the last decade, non-intrusive surrogate modeling for reservoir simulations has increased dramatically in popularity due to recent advances in machine learning (ML) (Nwachukwu, 2018). Many ML algorithms are used to learn linear or non-linear behavior in reservoir systems to create surrogate models.

For the types of problems usually encountered, ML algorithms are grouped based on the level of supervision they require to train surrogate models. These groups include supervised and unsupervised learning. In supervised learning, the algorithm is fed with a set of labeled examples, i.e., inputs and their corresponding target (or output) values. The ML model is then trained using the examples to generate a simple function that can map a given input to its target value. Algorithms of this kind include Linear Regression (LR), Random Forests (RFs), Artificial Neural Networks(ANNs), etc. On the other hand, unsupervised learning algorithms train models using completely unlabeled examples, and in general, their main goal is to detect natural groupings in the dataset. Examples of such algorithms include k-Means, Hierarchical Clustering(HC) algorithms, etc.

Supervised ML-based surrogate models have been used to predict spatially-varying reservoir states such as field saturations and pressures, or well responses such as production rates and bottom-hole pressures under the water flooding strategy and different geological structures; see e.g., Nwachukwu et al. (2018); Nwachukwu (2018); Sampaio et al. (2009); Memon et al. (2014); Amini et al. (2012).

In general, to predict reservoir field quantity such as pressures or saturations, considering a two-phase flow, the trained supervised ML-model is a vector-valued map of the form;

$$\text{ML-model} : \mathbb{R}^{N_u} \to \mathbb{R}^{2N_g \cdot N_t}. \tag{5.1}$$

To every input control $\mathbf{U} \in \mathbb{R}^{N_u}$ the model predicts the reservoir field quantity in the N_g- grid cells respectively for each fluid present over N_t time steps. That's, each component of the model target corresponds to a fluid field quantity in a given grid cell per time. The captured information can then be used to estimate the well's output of the reservoir. To fit such a model, especially in cases where the reservoir is discretized into a large number of grid cells, will require a substantial amount of labeled data to capture the entire reservoir behavior. In such a situation, the training complexity can be reduced by using the Principal Component Analysis (PCA) to project the high dimensional input or output space to a much lower-dimensional one, and then perform training with respect to the low dimensional space. Instead of using the PCA approach, one can also consider targeting grid cells, where the geological properties of interest are more influential.

5.3 General application of non-intrusive ROM in EOR optimization

Mainly, in the context of water flooding optimization, the highlighted ROM techniques above have been used to build surrogate models to predict the outcome of FOM simulation as per the reservoir responses and hence outputs for NPV evaluations. Also, several procedures on how to utilize the surrogate model to arrive at the optimum solutions have been extensively discussed see, e.g., (Nwachukwu, 2018; Durlofsky, 2010). Although this optimization approach still involves an indirect evaluation of the objective function at each input control, the speedup in overall computation is found to be very attractive.

Instead of providing surrogate models for indirect objective function evaluations, Ahmadi (2015); Cheraghi et al. (2021); Saberi et al. (2021) used ANN to directly approximate the costly input-output objective function in an EOR flooding settings. Here, the main focus is on accelerating the evaluation of objective function without insight on how to use surrogate to solve the underlying EOR flow problem. Golzari et al. (2015) provides an algorithm to obtain a global surrogate model in a genetic optimization algorithm. In their formulation, the global model is computed a priori before the optimization process begins.

Considering reservoir complexity associated with EOR methods, it may be impossible to construct a surrogate model that is accurate for the entire objective function input space. Moreover, if such global model is possible, it would need extremely huge amount of time for gathering the training dataset and finding the surrogate parameters. This would translate to losing the focus of accelerating computation.

To avoid the problem of constructing a globally accurate surrogate model, this study utilizes Deep Neural Networks (DNNs) to directly and locally build a surrogate model for the computationally expensive scalar-output EOR objective function (3.8) defined by (3.3). More so, the procedures on how to appropriately adapt or enrich the surrogate model while traversing the solution space during EOR optimization are proposed.

5.4 Deep neural networks

The concept and working principle of ANN follow from the biological neural system, where several neurons (approximately 10^{11}) are interconnected and perform specific tasks from their own experience (Zurada, 1992). A neuron is a processing unit, which takes some input data to give only one output, for instance, see Figure 5.3. Here, each input x_i is weighted by a factor w_i and the weighted sum of inputs with a bias factor b is computed as $a = b + \sum_i w_i x_i$. Then a activation function f is applied to the result a to get the neuronal output $f(a)$. Usually, the activation function is a simple mathematical function that decides whether a neuron is activated or not. It defines how the weighted sum of the inputs is transformed into an output from a node or nodes in a layer of the network. Some commonly used activation functions are identity function, sigmoid function, Tanh function, Rectified Linear Unit (ReLU) function, etc (Chakraverty and Jeswal, 2021).

The ANNs are built by interconnecting the neurons in layers. In general, the

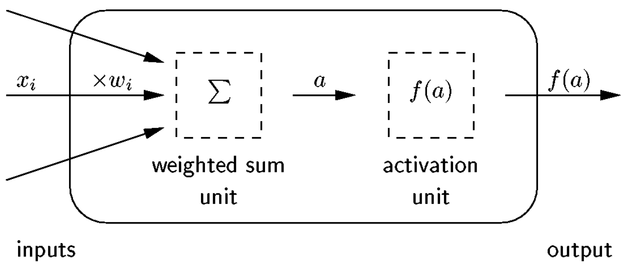


Figure 5.3: The neuron (Fukunaga, 1990).

ANN comprises of three layers, namely input, hidden, and output. The input layer interfaces with the external environment that feeds the the ANN model with input data. The hidden layer is the intermediate layer between the input and output layers. ANNs with more than one hidden layers are called DNNs, see e.g., Figure 5.4. The output layer is the last layer that gathers all the information to produce the desired model output (Chakraverty and Jeswal, 2021; Fukunaga, 1990).

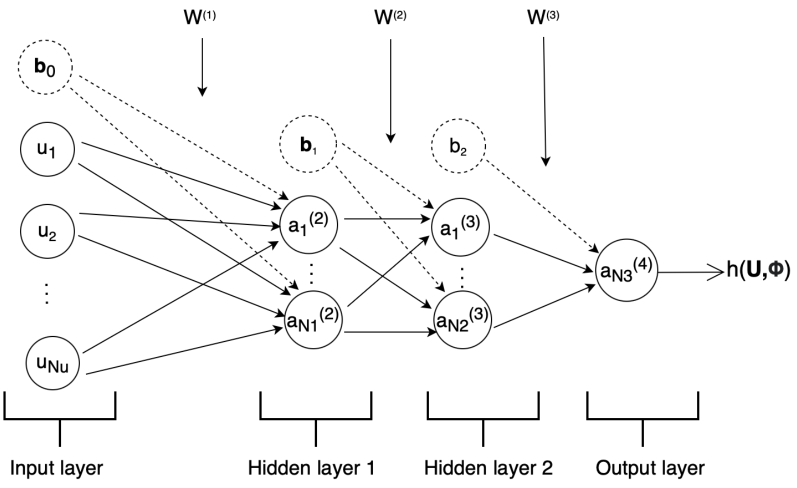


Figure 5.4: Feedforward neural network for the EOR objective function (3.3).

In general, the goal of DNNs is to approximate some arbitrary function

$$\begin{aligned} \mathbf{f}^* : \mathbb{R}^{N_{in}} &\longrightarrow \mathbb{R}^{N_{out}} \\ \mathbf{U} &\longmapsto \mathbf{f}^*(\mathbf{U}) \end{aligned} \tag{5.2}$$

where $N_{in} \in \mathbb{N}$ and $N_{out} \in \mathbb{N}$ denote the dimension of the function input and output. Here, a feedforward DNNs defines a map $\mathbf{h}(\mathbf{U}; \mathbf{\Phi})$ and learns the value of the parameter $\mathbf{\Phi}$ that results in the best function approximation of $\mathbf{f}^*$ (Goodfellow et al., 2016). The DNN is called feedforward because information flows through the function being evaluated from $\mathbf{U}$ (in the input layer) through the intermediate computations (in the hidden layers) used to define $\mathbf{h}$ without feedback. That's, no cyclic flow of information.

By setting $\mathbf{f}^*$ in (5.2) to the EOR objective function (3.8), $N_{in} = N_u$ and $N_{out} = 1$. The mathematical formulation of the scalar-valued feedforward DNN model approximation of J denoted by $h(\mathbf{U}, \mathbf{\Phi})$ using the neural network in Figure 5.4 is as follow.

According to the feedforward DNN in Figure 5.4, let $L = 4$ be the number of layers; $N_u = N_0, N_1, N_2$, and $N_{L-1} = 1$ be the number of neurons in each layer. The matrix of weights connecting the neurons in layer i to $i + 1$ is denoted by $\mathbf{W}^{(i)} \in \mathbb{R}^{N_{i-1} \times N_i}, i = 1, 2, 3$, and $\mathbf{b}_{i-1} \in \mathbb{R}^{N_i}, i = 1, 2, .., L - 1$ are the vector of biases in the network (note that, in this example $b_2 \in \mathbb{R}$). Hence, the model parameters to learn is given by $\mathbf{\Phi} := \{(\mathbf{W}^{(i)}, \mathbf{b}_{i-1})\}_{i=1}^{L-1}$. Suppose that an $m-$th neuron in layer i is connected to an $n-$th neuron in layer $i + 1$, the weight of this connection is denoted as $W_{nm}^{(i)}$ in the weight matrix $\mathbf{W}^{(i)}$ for each $i = 1, 2, .., L - 1$. For a given input $\mathbf{U} = \{u_i\}_{i=1}^{N_u}$, and suitable activation function $g : \mathbb{R} \longrightarrow \mathbb{R}$, the outputs $\{a_j^{(i+1)}\}_{j=1}^{N_i}$, for each $i = 1, 2, .., L - 1$ are computed as follows:

$$a_1^{(2)} = g(\sum_{m=1}^{N_u} W_{1m}^{(1)} u_m + W_{1b}^{(1)} b_{0,1}), \tag{5.3}$$

$$\vdots$$

$$a_{N_1}^{(2)} = g(\sum_{m=1}^{N_u} W_{N_1 m}^{(1)} u_m + W_{N_1 b}^{(1)} b_{0,N_1}), \tag{5.4}$$

$$a_1^{(3)} = g(\sum_{m=1}^{N_1} W_{1m}^{(2)} a_m^{(2)} + W_{1b}^{(2)} b_{1,1}), \tag{5.5}$$

$$\vdots$$

$$a_{N_2}^{(3)} = g(\sum_{m=1}^{N_1} W_{N_2 m}^{(2)} a_m^{(2)} + W_{N_2 b}^{(1)} b_{1,N_2}), \tag{5.6}$$

and finally,

$$h(\mathbf{U},\mathbf{\Phi}) := a^{(4)}_{N_3} = g(\sum_{m=1}^{N_2} W^{(3)}_{N_3 m} a^{(3)}_m + W^{(3)}_{N_3 b} b_2). \tag{5.7}$$

Here, $\mathbf{B}_w = (W^{(1)}_{1b}, W^{(1)}_{2b}, ..., W^{(2)}_{N_1 b}, W^{(2)}_{2b}, ..., W^{(2)}_{N_2 b}, W^{(3)}_{N_3 b})$ corresponds to the vector of weights for the biases. In this study, the components of $\mathbf{B}_w$ are set to 1, see **Paper IV**. For suitable model parameter $\mathbf{\Phi}$, the resulting DNNs model (5.7) becomes the desired function approximation of J. The process of finding suitable $\mathbf{\Phi}$ such that $h(\mathbf{U},\mathbf{\Phi}) \approx J(\mathbf{U})$ is called training of the neural network. It is a minimization problem where the objective function is some loss function measuring the deviation of $h(\mathbf{U}_\kappa,\mathbf{\Phi})$ from the exact value $J(\mathbf{U}_\kappa)$ for a given set of input-output training data $\{\mathbf{U}_\kappa, J(\mathbf{U}_\kappa)\}_{\kappa\in\mathcal{T}}$ and validation data $\{\mathbf{U}_\kappa, J(\mathbf{U}_\kappa)\}_{\kappa\in\mathcal{V}}$, where $\mathcal{T}$ and $\mathcal{V}$ are indexing sets. Moreover, the training and validation data are disjoint and have the same structure. The process of determining how well the model accurately predicts the output of previously unseen data is called validation of the neural network. The training procedures of present study can be found in **PAPER IV**.

For the DNN minimization problem, the gradient information of the loss function is computed using the Backpropagation method, for detailed explanation see Appendix A.3.

In **Paper IV**, we consider approximating two different structures of the EOR objective function (3.3), namely scalar- and vector-valued (both with a fixed geological property). The scalar-valued version remains the same as in (3.3). In this case, the special form of the scalar output feedforward DNNs in Figure 5.4 (i.e., components of $\mathbf{B}_w$ equal 1) is used to build a surrogate model, here denoted as $\hat{J}_s$. The vector-valued version is simply considering the same input space as in (3.3) but with vector output of the objective function without the discount factor. That's

$$\hat{\mathbf{J}}_v : \mathbb{R}^{N_t} \longrightarrow \mathbb{R}^{N_t} \tag{5.8}$$

$$\mathbf{U} \longmapsto \hat{\mathbf{J}}_v := \{J_i(\mathbf{U})\}_{i=1}^{N_t} \tag{5.9}$$

(see Section 3 for the definition of J_i and N_t). Each component of the output corresponds to the objective function value in the $i-$th simulation time. Here, the feedforward DNNs model with components of $\mathbf{B}_w$ set to 1 has a vector output of size N_t, i.e., $\mathbf{h}(\mathbf{U},\mathbf{\Phi}) \in \mathbb{R}^{N_t}$. Hence, the true object function value approximation at any given input using the DNNs model is obtained by multiplying the vector output with the vector of discount factors in the simulation time steps.

Furthermore, **PAPER IV** proposes a novel methodology of locally building and adapting the DNN models of the expensive EOR objective function (3.3) during the course of solving the EOR optimization problem (3.3)-(3.6). Each successful update step in FOM-based optimization method is referred to as outer iteration. For a given solution initialization, the gradient data of the FOM objective function J at this initial point is used to locally build a DNN models $\hat{J}_s$ or $\hat{\mathbf{J}}_v$ (depending on the structure of J).

The optimization routine in Algorithm 2 is then performed with a surrogate replacing the FOM-based J. Here, each update step is called an inner iteration. The quality of the solution obtained after convergence is checked using some FOM-based criteria and ensures that the optimum obtain by the surrogate is approximately a local optimum of true objective function (3.3). Should the checking indicates a tendency for solution improvement or poor approximation, the DNN model is adapted using FOM gradient quantities and perform the optimization routine with the newly adapted model. This process is repeated until convergence.

With a polymer EOR optimization problem on a heterogeneous 2D reservoir model, the proposed Adaptive-ML optimization technique finds improved optimal solutions with higher NPVs compared to the FOM-based optimization procedures (unlike the POD-based ROM optimization procedures, which leads to a sub-optimal solution with lesser NPV compared to the FOM-based optimization method, see Figure 5.2). Moreover, the speed up in computation using our proposed adaptive method is approximately up to a factor of 14.

6 Conclusion

The EOR models are essential for studying EOR effects to improve oil recovery. The models can appropriately predict the reservoir dynamic state and hence the output in terms of fluid production to a given EOR strategy at a point in time. This thesis studied the EOR modeling process for polymer, Smart water, and solvent flooding useful for the constrained optimization, value quantification, and ranking of EOR strategies on relevant oil reservoirs. Furthermore, an efficient and non-intrusive robust method for computing the gradient of the EOR NPV function and a technique for the appropriate handling of the underlying constraints during the EOR optimization procedures are demonstrated. The first section of present chapter summarizes each chapter of this thesis, while the second section presents the future outlook of study.

6.1 Summary

EOR modeling and open porous media: This chapter presents the reservoir modeling for chemical and gas EOR methods. Initially, the chapter illustrates the black-oil model describing the simultaneous flow of three phases (and three pseudo-components), the analysis of unknown primary variables in the black-oil model, and closure properties (the three phases include water, oil, and gas). Then, it presents EOR models, which are extensions of the black-oil model. The black-oil model consists of three partial differential equations, each representing the conservation of mass of a phase. Here, mass is transported at a velocity measured by Darcy's law. The unknown primary variables such as phase pressure and saturation measure the reservoir states, which change depending on the input control per time. The closure properties include the saturation and capillary relations of states.

The chemical EOR models describe polymer and Smart water flooding, while the gas EOR model describes solvent flooding for given reservoir properties. Each model is an extension of the black-oil model by appropriate conservation equation for the additional component $\alpha \in \{\text{polymer, Smart water, solvent}\}$ and closure properties. The spatial and time discretization of the black-oil and EOR model equations lead to non-linear discrete-time algebraic equations whose solution using the Newton-Raphson method gives the reservoir discrete-time states. The discretization and algebraic solution toolbox for this purpose are in the OPM-flow simulator. Given an EOR setup, the reservoir states, and hence, reservoir outputs using the appropriate well model, can be computed for an EOR input control strategy.

EOR optimization: This chapter introduces the mathematical formulation of the constrained optimization problem associated with EOR methods useful for their value quantification on relevant oil fields. Here, the objective function is the reservoir NPV function J, which measures the economic value of the injected and produced fluids discounted annually. Indirectly, it is defined on controllable input vector $\mathbf{U}$ with components such as oil and water rate, EOR gas or chemical concentration, etc. For every $\mathbf{U} \in \mathbb{R}^{N_w \cdot N_t}$, assume that $\mathbf{X}$ is the reservoir state function define by

$$\begin{aligned} \mathbf{X} : \mathbb{R}^{N_w \cdot N_t} &\longrightarrow \mathbb{R}^{N_{ph} \cdot N_{sf} \cdot N_{gc} \cdot N_t} \\ \mathbf{U} &\longmapsto \mathbf{X}(\mathbf{U}), \end{aligned} \tag{6.1}$$

where N_{ph}, N_{sf}, N_{gc}, and N_t are the number of phases, number of state fields, and number of grid cells in the reservoir. Here, the output of $\mathbf{X}$ is obtained by solving the underlying reservoir flow equations. For a suitable wells function $\mathcal{W}_e$ defined by

$$\begin{aligned} \mathcal{W}_e : \mathbb{R}^{N_{ph} \cdot N_{sf} \cdot N_{gc} \cdot N_t} &\longrightarrow \mathbb{R}^{N_w \cdot N_t} \\ \mathbf{X}(\mathbf{U}) &\longmapsto \mathbf{Y} := \mathcal{W}_e(\mathbf{X}(\mathbf{U})), \end{aligned} \tag{6.2}$$

which measure the reservoir output in terms of fluid productions at the production wells, the NPV function J quantifies the economic value of the reservoir output $\mathbf{Y}$ as a result of the input vector $\mathbf{U} \in \mathbb{R}^{N_w \cdot N_t}$. Therefore, the EOR optimization problem is to find the best $\mathbf{U}$ that maximizes the economic value of the reservoir outputs. Because there are limited injection and production facilities, possible optimization constraints defined on the input variables are presented.

Gradient-based optimization method: In this chapter, the gradient-based solution method for high-dimensional constrained EOR optimization problems is presented. First, different methods for computing the gradient of the objective function, namely the adjoint and stochastic methods, are explained. The adjoint method accurately computes the gradient of J but requires access to the reservoir flow equations. Moreover, the solution method utilizing the adjoint method is prone to finding local optimum points. The stochastic methods (e.g., the EnOpt method), on the other hand, utilize input control perturbations and their function evaluations to mathematically approximate the gradient function. They are non-intrusive, i.e., require no access to the flow equations.

Further, this chapter introduces the general formulation of the EnOpt method for gradient computation in deterministic and robust settings. The deterministic case assumes that the reservoir has known geological properties, while the robust case considers geological uncertainty quantification. Of interest to this study is the quantification of the economic benefits of EOR strategies (input controls) in the robust setting. Hence the accuracy of the EnOpt method, in this case, is analyzed. Using the robust EnOpt gradient and suitable constraint handling technique, this chapter presents an optimization algorithm for finding the optimal solution to the REORO problem. In this case, the algorithm is based on the inexact line search method. The constraint handling technique uses some admissible set equivalent (in the sense of some linear transformation) to the original solution domain. The optimization results are used to quantify and rank the value of EOR methods for different oil reservoirs designed to mimic a North Sea field. Moreover, this chapter illustrates the impact on the quality of gradient computed by the EnOpt method when using the traditional constraint handling technique, which leads to an inefficient solution method. Because of this, a new approach for the appropriate handling of underlying constraints when solving the EOR problem is presented. The technique utilizes the exterior penalty method. Here the constrained problem is converted into a sequence of unconstrained subproblems using penalty parameters. By solving sub-problems sequentially using appropriate unconstrained method, the solutions of sub-problems converge to the optimal solution of the EOR problem as the penalty parameter increases.

Model order reduction for EOR optimization: For the derivation of an efficient and improved solution method for EOR optimization problems, this chapter illustrates possible ROM strategies to accelerate the computationally expensive objective function (3.3). The evaluation of the EOR objective function relies on costly reservoir simulations, especially with complicated reservoirs. Intrusive ROM techniques approximate the expensive FOM simulations using POD and TPWL-based POD to obtain basis functions from the snapshot matrix. Here, the subsequent solutions of the FOM reservoir simulation are then represented as linear combinations of the basis functions. In contrast, non-intrusive ROM methods based on machine learning algorithms for reservoir modeling are presented. In general, simple mathematical functions are trained to model the linear and non-linear reservoir behaviors. These ROM methods combined with optimization algorithms have been used to solve water flooding problems.

Also, this chapter explores applications of non-intrusive ROM methods for EOR problems. Here, major researches are concerned with global model approximation or computation acceleration of the expensive input-output objective function of the underlying EOR problems. For instance, the problem with using a global model in the course of the optimization routines is illustrated. Instead of global approximation, this chapter presents a new method for local and direct approximation of the EOR objective function (3.3) using feedforward DNNs. Also, procedures (certified using FOM quantities) on how to adapt or enrich the surrogate model during the optimization routine for the EOR optimization problems (3.3)-(3.6) are illustrated.

6.2 Research outlook

One main goal of this study is to optimize the EOR objective function (4.12) formulated based on the EOR models presented in Chapter 2. The proposed EOR models have been successfully used to simulate EOR effects of polymer, Smart water, and solvent flooding on oil recovery. The mathematical formulation of EOR models are based on the assumptions that all hydrocarbon species can be lumped together as two components, namely gas and oil, with fixed chemical composition at surface conditions (see Section 2.1). Here, no account for cases where more than two different compositions or species of reservoir hydrocarbon are required. For such cases, there is need to consider EOR models, which are extensions of the compositional model (Coats, 1980) instead of the black-oil model. In this regard, the proposed EOR optimization problem formulation and solution method in this study remain applicable.

The focus of this study has been on the robust optimization and value quantification of one EOR method over the lifespan of a given oil reservoir at a time. As future work, it will be interesting to consider optimizing scenarios with combined EOR methods and compare results with individual EOR strategies. Moreover, this study utilizes mostly bound constraints on the individual EOR input variables. Therefore, the investigation of constraints on combinations of input variables of the same type is an important possibility.

The new constraint technique proposed by this study has been extensively used for reservoir optimization problems with bound constraints. The reservoirs are assumed to have known geological properties. In addition, for analytical problems with linear constraints, the method proves to be more efficient. As future work, one can consider the technique for robust constrained optimization problems with mixed or more complex non-linear constraints.

Finally, this study has successfully developed a new efficient optimization algorithm that utilizes the surrogate of the objective function (3.3) to find the optimum solution to the underlying EOR optimization. However, a case where the geological properties are known is applicable with methodology. In practice and as discussed in Chapter 1, these geological parameters are usually unknown, and because of this, it is imperative to consider an ensemble of geologies to quantify this uncertainty. As future work, this algorithm can be extended to the robust EOR optimization problem, where surrogates are locally created for the objective function (4.12).

References

Ahmadi, M. A. (2015). Developing a robust surrogate model of chemical flooding based on the artificial neural network for enhanced oil recovery implications. *Mathematical Problems in Engineering 2015*.

Alfazazi, U., W. AlAmeri, and M. R. Hashmet (2019). Experimental investigation of polymer flooding with low-salinity preconditioning of high temperature–high-salinity carbonate reservoir. *Journal of Petroleum Exploration and Production Technology 9*(2), 1517–1530.

Alizadeh Nomeli, M. and A. Riaz (2013). Reactive transport modeling of CO2 inside a fractured rock: Implications of mass transfer and storage capacity. In *APS Division of Fluid Dynamics Meeting Abstracts*, pp. M26–004.

Amini, S., S. Mohaghegh, R. Gaskari, and G. Bromhal (2012). Uncertainty analysis of a CO2 sequestration project using surrogate reservoir modeling technique. In *SPE Western Regional Meeting*. OnePetro.

Arouri, Y. and M. Sayyafzadeh (2020). An accelerated gradient algorithm for well control optimization. *Journal of Petroleum Science and Engineering 190*, 106872.

Astrid, P., G. Papaioannou, J. C. Vink, and J. Jansen (2011). Pressure preconditioning using proper orthogonal decomposition. In *SPE Reservoir Simulation Symposium*. OnePetro.

Baker, R. (1998). *A primer of offshore operations*. University of Texas at Austin Petroleum.

Bao, K., K.-A. Lie, O. Møyner, and M. Liu (2017). Fully implicit simulation of polymer flooding with mrst. *Computational Geosciences 21*(5), 1219–1244.

Baxendale, D., A. F. Rasmussen, A. B. Rustad, T. Skille, and T. H. Sandve (2021). Opm flow documentation manual manual. *Open Porous Media Initiative*.

Brouwer, D. R. and J. Jansen (2002). Dynamic optimization of water flooding with smart wells using optimal control theory. In *European Petroleum Conference*. Society of Petroleum Engineers.

Cardoso, M. A. and L. J. Durlofsky (2010). Linearized reduced-order models for subsurface flow simulation. *Journal of Computational Physics 229*(3), 681–700.

Cardoso, M. A., L. J. Durlofsky, and P. Sarma (2009). Development and application of reduced-order modeling procedures for subsurface flow simulation. *International journal for numerical methods in engineering 77*(9), 1322–1350.

Chakraverty, S. and S. K. Jeswal (2021). *Applied Artificial Neural Network Methods for Engineers and Scientists: Solving Algebraic Equations*. World Scientific.

Chase, C. A. and M. R. Todd (1984). Numerical simulation of CO2 flood performance (includes associated papers 13950 and 13964). *Society of Petroleum Engineers Journal 24*(06), 597–605.

Chen, G., K. Zhang, X. Xue, L. Zhang, J. Yao, H. Sun, L. Fan, and Y. Yang (2020). Surrogate-assisted evolutionary algorithm with dimensionality reduction method for water flooding production optimization. *Journal of Petroleum Science and Engineering 185*, 106633.

Chen, Y. and D. S. Oliver (2010). Ensemble-based closed-loop optimization applied to brugge field. *SPE Reservoir Evaluation & Engineering 13*(01), 56–71.

Chen, Y., D. S. Oliver, and D. Zhang (2009). Efficient ensemble-based closed-loop production optimization. *SPE Journal 14*(04), 634–645.

Chen, Z. (2007). *Reservoir simulation: mathematical techniques in oil recovery*. SIAM.

Cheraghi, Y., S. Kord, and V. Mashayekhizadeh (2021). Application of machine learning techniques for selecting the most suitable enhanced oil recovery method; challenges and opportunities. *Journal of Petroleum Science and Engineering 205*, 108761.

Coats, K. H. (1980). An equation of state compositional model. *Society of Petroleum Engineers Journal 20*(05), 363–376.

Davidon, W. (1959). Variable metric method for minimization.

Do, S. T. and A. C. Reynolds (2013). Theoretical connections between optimization algorithms based on an approximate gradient. *Computational Geosciences 17*(6), 959–973.

Dudek, J., D. Janiga, and P. Wojnarowski (2021). Optimization of CO2-EOR process management in polish mature reservoirs using smart well technology. *Journal of Petroleum Science and Engineering 197*, 108060.

Durlofsky, L. (2010). Use of reduced-order modeling procedures for production optimization. *SPE Journal 15*(02), 426–435.

Eberhart, R. and J. Kennedy (1995). Particle swarm optimization. In *Proceedings of the IEEE international conference on neural networks*, Volume 4, pp. 1942–1948. Citeseer.

Fani, M., H. Al-Hadrami, P. Pourafshary, G. Vakili-Nezhaad, and N. Mosavat (2018). Optimization of smart water flooding in carbonate reservoir. In *Abu Dhabi International Petroleum Exhibition & Conference*. Society of Petroleum Engineers.

Fonseca, R., E. Della Rossa, A. Emerick, R. Hanea, and J. Jansen (2018). Overview of the olympus field development optimization challenge. In *ECMOR XVI-16th European Conference on the Mathematics of Oil Recovery*, Volume 2018, pp. 1–10. European Association of Geoscientists & Engineers.

Fonseca, R., S. Kahrobaei, L. Van Gastel, O. Leeuwenburgh, and J. Jansen (2015). Quantification of the impact of ensemble size on the quality of an ensemble gradient using principles of hypothesis testing. In *SPE Reservoir Simulation Symposium*. OnePetro.

Fonseca, R., O. Leeuwenburgh, P. Van den Hof, and J.-D. Jansen (2014). Improving the ensemble-optimization method through covariance-matrix adaptation. *SPE Journal 20*(01), 155–168.

Fonseca, R. R.-M., B. Chen, J. D. Jansen, and A. Reynolds (2017). A stochastic simplex approximate gradient (StoSAG) for optimization under uncertainty. *International Journal for Numerical Methods in Engineering 109*(13), 1756–1776.

Fukunaga, K. (1990). The artificial neural network book.

Goldberg, D. E. (1989). Genetic algorithms in search, optimization, and machine learning. addison. *Reading*.

Golzari, A., M. H. Sefat, and S. Jamshidi (2015). Development of an adaptive surrogate model for production optimization. *Journal of petroleum Science and Engineering 133*, 677–688.

Goodfellow, I., Y. Bengio, and A. Courville (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Heijn, T., R. Markovinovic, and J.-D. Jansen (2004). Generation of low-order reservoir models using system-theoretical concepts. *SPE Journal 9*(02), 202–218.

Higham, N. J. (2008). *Functions of matrices: theory and computation*. SIAM.

Holmes, J. (1983). Enhancements to the strongly coupled, fully implicit well model: wellbore crossflow modeling and collective well control. In *SPE Reservoir Simulation Symposium*. OnePetro.

Holmes, J., T. Barkve, and O. Lund (1998). Application of a multisegment well model to simulate flow in advanced wells. In *European petroleum conference*. OnePetro.

Holmes, P., J. L. Lumley, G. Berkooz, and C. W. Rowley (2012). *Turbulence, coherent structures, dynamical systems and symmetry*. Cambridge university press.

Hou, J., K. Zhou, X.-S. Zhang, X.-D. Kang, and H. Xie (2015). A review of closed-loop reservoir management. *Petroleum Science 12*(1), 114–128.

Islam, J., P. M. Vasant, B. M. Negash, M. B. Laruccia, M. Myint, and J. Watada (2020). A holistic review on artificial intelligence techniques for well placement optimization problem. *Advances in Engineering Software 141*, 102767.

Jain, M. K. (2003). *Numerical methods for scientific and engineering computation*. New Age International.

Jakupsstovu, S., D. Zhou, J. Kamath, L. Durlofsky, and E. H. Stenby (2001). Upscaling of miscible displacement processes. In *Proceedings of the 6th Nordic Symposium on Petrophysics*, pp. 15–16.

Janiga, D., R. Czarnota, E. Kuk, J. Stopa, and P. Wojnarowski (2020). Measurement of oil-CO2 diffusion coefficient using pulse-echo method for pressure-volume decay approach under reservoir conditions. *Journal of Petroleum Science and Engineering 185*, 106636.

Jansen, J. (2011a). Adjoint-based optimization of multi-phase flow through porous media - a review. *Computers & Fluids 46*(1), 40 – 51. 10th ICFD Conference Series on Numerical Methods for Fluid Dynamics (ICFD 2010).

Jansen, J. D. (2011b). Adjoint-based optimization of multi-phase flow through porous media–a review. *Computers & Fluids 46*(1), 40–51.

Jansen, J.-D., O. H. Bosgra, and P. M. Van den Hof (2008). Model-based control of multiphase flow in subsurface oil reservoirs. *Journal of Process Control 18*(9), 846–855.

Jansen, J.-D., R. Brouwer, and S. G. Douma (2009). Closed loop reservoir management. In *SPE reservoir simulation symposium*. Society of Petroleum Engineers.

Jesmani, M., B. Jafarpour, M. C. Bellout, and B. Foss (2020). A reduced random sampling strategy for fast robust well placement optimization. *Journal of Petroleum Science and Engineering 184*, 106414.

Jia, X., K. Ma, Y. Liu, B. Liu, J. Zhang, and Y. Li (2013). Enhance heavy oil recovery by in-situ carbon dioxide generation and application in china offshore oilfield. In *SPE Enhanced Oil Recovery Conference*. Society of Petroleum Engineers.

Jung, S., K. Lee, C. Park, and J. Choe (2018). Ensemble-based data assimilation in reservoir characterization: A review. *Energies 11*(2), 445.

Kaleta, M. P., R. G. Hanea, A. W. Heemink, and J.-D. Jansen (2011). Model-reduced gradient-based history matching. *Computational Geosciences 15*(1), 135–153.

Kiefer, J. (1957). Optimum sequential search and approximation methods under minimum regularity assumptions. *Journal of the Society for Industrial and Applied Mathematics 5*(3), 105–136.

Kraaijevanger, J., P. Egberts, J. Valstar, and H. Buurman (2007). Optimal waterflood design using the adjoint method. In *SPE Reservoir Simulation Symposium*. OnePetro.

Lee, A. and J. Aronofsky (1958). A linear programming model for scheduling crude oil production. *Journal of Petroleum Technology 10*(07), 51–54.

Lei, Y., S. Li, X. Zhang, Q. Zhang, and L. Guo (2012). Optimal control of polymer flooding based on maximum principle. *Journal of Applied Mathematics 2012*.

Li, G. and A. C. Reynolds (2011). Uncertainty quantification of reservoir performance predictions using a stochastic optimization algorithm. *Computational Geosciences 15*(3), 451–462.

Li-xin, G. X.-j. W. and Y. Jian-jun (2005). Optimization of operation plan for water injection system in oilfield using hybrid genetic algorithm [j]. *Acta Petrolei Sinica 3*.

Liu, D. C. and J. Nocedal (1989). On the limited memory bfgs method for large scale optimization. *Mathematical programming 45*(1), 503–528.

Liu, X. and A. C. Reynolds (2014). Gradient-based multiobjective optimization with applications to waterflooding optimization. In *ECMOR XIV-14th European Conference on the Mathematics of Oil Recovery*, Volume 2014, pp. 1–21. European Association of Geoscientists & Engineers.

Lorentzen, R. J., A. Berg, G. Nævdal, and E. H. Vefring (2006). A new approach for dynamic optimization of water flooding problems. In *Intelligent Energy Conference and Exhibition*. Society of Petroleum Engineers.

Lu, R., F. Forouzanfar, and A. C. Reynolds (2017). Bi-objective optimization of well placement and controls using StoSAG. In *SPE reservoir simulation conference*. OnePetro.

Lyons, W. (2009). *Working guide to reservoir engineering*. Gulf professional publishing.

Mehos, G. J. and W. F. Ramirez (1989). Use of optimal control theory to optimize carbon dioxide miscible-flooding enhanced oil recovery. *Journal of Petroleum Science and Engineering 2*(4), 247–260.

Memon, P. Q., S.-P. Yong, W. Pao, and P. J. Sean (2014). Surrogate reservoir modeling-prediction of bottom-hole flowing pressure using radial basis neural network. In *2014 Science and Information Conference*, pp. 499–504. IEEE.

Milk, R., S. Rave, and F. Schindler (2016). pyMOR–generic algorithms and interfaces for model order reduction. *SIAM Journal on Scientific Computing 38*(5), S194–S216.

Muggeridge, A., A. Cockin, K. Webb, H. Frampton, I. Collins, T. Moulds, and P. Salino (2014). Recovery rates, enhanced oil recovery and technological limits. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 372*(2006), 20120320.

Nævdal, G., D. R. Brouwer, and J.-D. Jansen (2006). Waterflooding using closed-loop control. *Computational Geosciences 10*(1), 37–60.

Neumann, A. W., R. David, and Y. Zuo (2010). *Applied surface thermodynamics*, Volume 151. CRC press.

Niu, J., Q. Liu, J. Lv, and B. Peng (2020). Review on microbial enhanced oil recovery: Mechanisms, modeling and field trials. *Journal of Petroleum Science and Engineering*, 107350.

Nocedal, J. and S. J. Wright (2006). *Numerical Optimization* (Second ed.). Springer.

Norwegian-Petroleum-Directorate (2019). Petroleum reources report on the norwegian continental shelf. `https://www.npd.no/en/facts/publications/reports2/resource-report/resourc-report-2019/fields/`.

Nwachukwu, A., H. Jeong, M. Pyrcz, and L. W. Lake (2018). Fast evaluation of well placements in heterogeneous reservoir models using machine learning. *Journal of Petroleum Science and Engineering 163*, 463–475.

Nwachukwu, C. (2018). *Machine learning solutions for reservoir characterization, management, and optimization*. Ph. D. thesis.

Ogbeiwi, P., Y. Aladeitan, and D. Udebhulu (2018). An approach to waterflood optimization: case study of the reservoir x. *Journal of Petroleum Exploration and Production Technology 8*(1), 271–289.

Patelli, E. and H. J. Pradlwarter (2010). Monte carlo gradient estimation in high dimensions. *International journal for numerical methods in engineering 81*(2), 172–188.

Powell, M. J. (1964). An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The computer journal 7*(2), 155–162.

Rao, S. S. (2019). *Engineering optimization: theory and practice*. John Wiley & Sons.

Rasmussen, A. F., T. H. Sandve, K. Bao, A. Lauser, J. Hove, B. Skaflestad, R. Klöfkorn, M. Blatt, A. B. Rustad, and O. Sævareid (2021). The open porous media flow reservoir simulator. *Computers & Mathematics with Applications 81*, 159–185.

Rewienski, M. and J. White (2003). A trajectory piecewise-linear approach to model order reduction and fast simulation of nonlinear circuits and micromachined devices. *IEEE Transactions on computer-aided design of integrated circuits and systems 22*(2), 155–170.

Ridwan, M. G., M. I. Kamil, M. Sanmurjana, A. M. Dehgati, P. Permadi, T. Marhaendrajana, and F. Hakiki (2020). Low salinity waterflooding: Surface roughening and pore size alteration implications. *Journal of Petroleum Science and Engineering 195*, 107868.

Saberi, H., E. Esmaeilnezhad, and H. J. Choi (2021). Artificial neural network to forecast enhanced oil recovery using hydrolyzed polyacrylamide in sandstone and carbonate reservoirs. *Polymers 13*(16), 2606.

Sadeed, A., Z. Tariq, A. N. Janjua, A. Asad, and M. E. Hossain (2018). Smart water flooding: an economic evaluation and optimization. In *SPE Kingdom of Saudi Arabia Annual Technical Symposium and Exhibition*. Society of Petroleum Engineers.

Sampaio, T. P., V. J. M. Ferreira Filho, and A. D. S. Neto (2009). An application of feed forward neural network as nonlinear proxies for use during the history matching phase. In *Latin American and Caribbean Petroleum Engineering Conference*. OnePetro.

Sarma, P., K. Aziz, and L. J. Durlofsky (2005). Implementation of adjoint solution for optimal control of smart wells. In *SPE reservoir simulation symposium*. Society of Petroleum Engineers.

Sarma, P. and W. Chen (2014). Improved estimation of the stochastic gradient with quasi-monte carlo methods. In *ECMOR XIV-14th European Conference on the Mathematics of Oil Recovery*, Volume 2014, pp. 1–25. European Association of Geoscientists & Engineers.

Sarma, P. and W. H. Chen (2008). Efficient well placement optimization with gradient-based algorithms and adjoint models. In *Intelligent energy conference and exhibition*. Society of Petroleum Engineers.

Sarma, P., W. H. Chen, L. J. Durlofsky, and K. Aziz (2006). Production optimization with adjoint models under nonlinear control-state path inequality constraints. In *Intelligent Energy Conference and Exhibition*. Society of Petroleum Engineers.

Schlegel, M. and B. R. Noack (2015). On long-term boundedness of galerkin models. *Journal of Fluid Mechanics 765*, 325–352.

Schlumberger, A. (2010). Eclipse technical description. *Schlumberger Information Solutions*.

Sehbi, B. S., S. M. Frailey, and A. S. Lawal (2001). Analysis of factors affecting microscopic displacement efficiency in CO2 floods. In *SPE Permian Basin Oil and Gas Recovery Conference*. Society of Petroleum Engineers.

Semnani, A., M. Ostadhassan, Y. Xu, M. Sharifi, and B. Liu (2021). Joint optimization of constrained well placement and control parameters using teaching-learning based optimization and an inter-distance algorithm. *Journal of Petroleum Science and Engineering 203*, 108652.

Shirangi, M. G. (2012). Applying machine learning algorithms to oil reservoir production optimization. In *Tech. Rep. Machine Learning Course Project Report*. Stanford University.

Snyman, J. A. and D. N. Wilke (2005). *Practical mathematical optimization*. Springer.

Stengel, R. F. (1994). *Optimal control and estimation*. Courier Corporation.

Sun, X.-h. and M.-h. Xu (2017). Optimal control of water flooding reservoir using proper orthogonal decomposition. *Journal of Computational and Applied Mathematics 320*, 120–137.

Svein, M. S. and J. Kleppe (1992). Recent advances in improved oil recovery methods for north sea sandstone reservoirs. *SPOR Monograph*.

Todd, M. and W. Longstaff (1972). The development, testing, and application of a numerical simulator for predicting miscible flood performance. *Journal of Petroleum Technology 24*(07), 874–882.

Van Doren, J., S. G. Douma, L. B. M. Wassing, J. Kraaijevanger, and A. H. De Zwart (2011). Adjoint-based optimization of polymer flooding. In *SPE Enhanced Oil Recovery Conference*. Society of Petroleum Engineers.

Van Doren, J. F., R. Markovinović, and J.-D. Jansen (2006). Reduced-order optimal control of water flooding using proper orthogonal decomposition. *Computational Geosciences 10*(1), 137–158.

Van Essen, G., M. Zandvliet, P. Van den Hof, O. Bosgra, and J.-D. Jansen (2009). Robust waterflooding optimization of multiple geological scenarios. *SPE Journal 14*(01), 202–210.

Völcker, C. (2011). Production optimization of oil reservoirs. *Kongens Lyngby, Denmark: Department of Informatics and Mathematical Modelling-Technical, University of Denmark*.

Walton, S., O. Hassan, and K. Morgan (2013). Reduced order modelling for unsteady fluid flow using proper orthogonal decomposition and radial basis functions. *Applied Mathematical Modelling 37*(20-21), 8930–8945.

Wang, G. C. (1980). Microscopic study of oil recovery by carbon dioxide. final report, october 1, 1978-august 31, 1980. Technical report, Alabama Univ., Tuscaloosa (USA). Dept. of Mineral Engineering.

Wang, Y., C. Liu, Y. Wang, and S. Zhou (2010). Optimization in oilfield water injection system based on algorithm of ant colony-particle swarm method. *J. Daqing Pet. Inst 2*, 014.

Xiangguo, L., C. Bao, X. Kun, C. Weijia, L. Yigang, Y. ZHANG, W. Xiaoyan, and J. ZHANG (2021). Enhanced oil recovery mechanisms of polymer flooding in a heterogeneous oil reservoir. *Petroleum Exploration and Development 48*(1), 169–178.

Xiao, D., F. Fang, C. Pain, I. Navon, and A. Muggeridge (2016). Non-intrusive reduced order modelling of waterflooding in geologically heterogeneous reservoirs. In *ECMOR XV-15th European conference on the mathematics of oil recovery*, pp. cp–494. European Association of Geoscientists & Engineers.

Xu, L., H. Zhao, Y. Li, L. Cao, X. Xie, X. Zhang, and Y. Li (2018). Production optimization of polymer flooding using improved monte carlo gradient approximation algorithm with constraints. *Journal of Circuits, Systems and Computers 27*(11), 1850167.

Yang, C. and X. Wang (2021). A steam injection distribution optimization method for sagd oil field using lstm and dynamic programming. *ISA transactions 110*, 198–212.

Yeten, B. (2003). *Optimum deployment of nonconventional wells*. Stanford University.

Yousef, A. and S. Ayirala (2014). Optimization study of a novel water-ionic technology for smart-waterflooding application in carbonate reservoirs. *Oil and Gas Facilities 3*(05), 72–82.

Yousef, A. A., S. Al-Saleh, and M. Al-Jawfi (2012). The impact of the injection water chemistry on oil recovery from carbonate reservoirs. In *SPE EOR Conference at Oil and Gas West Asia*. OnePetro.

Yousef, A. A., S. Al-Saleh, A. Al-Kaabi, and M. Al-Jawfi (2011). Laboratory investigation of the impact of injection-water salinity and ionic content on oil recovery from carbonate reservoirs. *SPE Reservoir Evaluation & Engineering 14*(05), 578–593.

Zandvliet, M., M. Handels, G. van Essen, R. Brouwer, and J.-D. Jansen (2008). Adjoint-based well-placement optimization under production constraints. *SPE Journal 13*(04), 392–399.

Zerkalov, G. (2015). Polymer flooding for enhanced oil recovery. *Stanford University*, 1–4.

Zhang, K., X. Zhang, W. Ni, L. Zhang, J. Yao, L. Li, and X. Yan (2016). Nonlinear constrained production optimization based on augmented lagrangian function and stochastic gradient. *Journal of Petroleum Science and Engineering 146*, 418–431.

Zhong, Z., A. Y. Sun, Y. Wang, and B. Ren (2020). Predicting field production rates for waterflooding using a machine learning-based proxy model. *Journal of Petroleum Science and Engineering 194*, 107574.

Zhou, K., J. Hou, X. Zhang, Q. Du, X. Kang, and S. Jiang (2013). Optimal control of polymer flooding based on simultaneous perturbation stochastic approximation method guided by finite difference gradient. *Computers & chemical engineering 55*, 40–49.

Zurada, J. (1992). *Introduction to artificial neural systems*. West Publishing Co.

A Appendix

A.1 Well model

The quantity Q_α in Equation (2.1) denotes well outflux/influx density (equivalently, the flow rate per unit volume or volumetric source/sink term) of pseudo component α, evaluated using semi-analytical models (Holmes, 1983; Holmes et al., 1998). Suppose that the reservoir domain is spatially discretized in such a way that the injection and production wells are located in different grid blocks as shown in Figure A.1. Each well is assumed to perforate a single grid block.

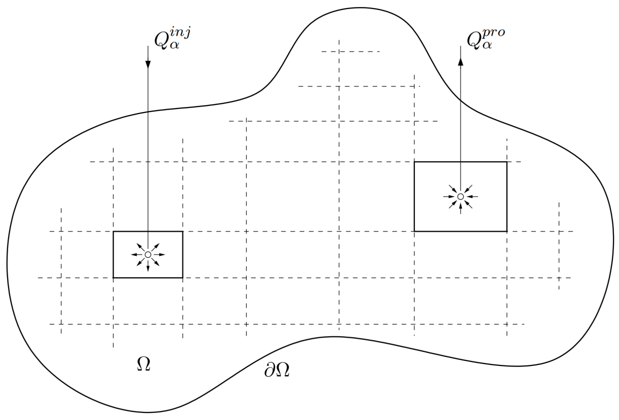


Figure A.1: Example of an injector and a producer located in two different grid blocks (Völcker, 2011).

In addition, if we assume a two-phase flow model of water and oil, the volumetric source/sink term $Q_\alpha := [Q_\alpha^{\text{inj}}, Q_\alpha^{\text{prod}}]$ at the wells can be modeled as follow. Let $\mathcal{N}$ be the set of grid blocks, $\mathcal{I} \subset \mathcal{N}$ be the set of grid blocks penetrated by the injection wells, and $\mathcal{P} \subset \mathcal{N}$ be the set of grid blocks containing the production wells. Here, the injectors are controlled by volumetric injection rates, and producers are controlled by bottom hole pressures (BHP)[1].

[1]Bottom hole pressure is the pressure inside the well at reservoir depth (Völcker, 2011).

A.1.1 Injection wells

For an injector at grid block $i \in \mathcal{I}$, the source term $Q_{\alpha,i}^{\text{inj}}$ (in kg/(m^{3}d)) can be directly controlled by setting

$$Q_{\alpha,i}^{\text{inj}} = \left(\frac{q_\alpha^{\text{inj}}}{V}\right)_i \qquad i \in \mathcal{I}, \tag{A.1}$$

where $q_{\alpha,i}^{\text{inj}}$ is the injected mass of phase α per unit time and V_i is the volume of $i-$th grid block. Assuming that the injected fluid is water, then

$$q_{w,i}^{\text{inj}} = (\rho_w \hat{q}_w^{\text{inj}})_i \quad i \in \mathcal{I} \tag{A.2}$$

$$q_{o,i}^{\text{inj}} = 0. \tag{A.3}$$

Here, $\hat{q}_{w,i}^{\text{inj}}$ denotes the volumetric injection rate (in m^3/d) of water into the grid block i.

A.1.2 Production wells

For grid block $i \in \mathcal{P}$, the sink term $Q_{\alpha,i}^{\text{prod}} := Q_{\alpha,i}^{\text{prod}}(p_\alpha, S_\alpha)$ measured in kg/(m^{3}d) cannot be directly controlled, since the produced fluid is a composition of oil and water. Here,

$$Q_{\alpha,i}^{\text{prod}} = \left(\frac{q_\alpha^{\text{inj}}}{V}\right)_i \qquad i \in \mathcal{I}, \tag{A.4}$$

where

$$q_{\alpha,i}^{\text{prod}} = -\left(\text{WI}\frac{\rho_\alpha k_{r\alpha}(S_\alpha)}{\mu_\alpha}(p_\alpha - p^{\text{bhp}})\right)_i. \tag{A.5}$$

Here, WI$_i$ denotes the well index, which indicates operational production wells and their interactions with the reservoir model. The Peaceman well index is expressed as

$$\text{WI}_i = \left(\frac{2\pi\sqrt{k_{11}k_{22}}h}{\ln(r_e/r_w) + \overline{s}}\right)_i, \tag{A.6}$$

where k_{11} and k_{22} are rock permeability components in the s_1 and s_2 coordinate axes respectively, h_i is the height of the well (height of grid block i in the s_3 direction), r_w is the wellbore radius, r_e is the drainage radius or the radial position, centered around the well, at which the pressure in the well block, calculated by the simulator, is the same as the pressure obtained by the semi-analytical model, and $\bar{s}$ is a constant factor added to match theoretical well productivity. Furthermore, p_i^{bhp} in (A.5) is the BHP of the production well in grid block i. For other well models suitable for reservoirs with three-phase flow and more complicated wells, see (Chen, 2007; Rasmussen et al., 2021).

A.2 Adjoint model equations and algorithm

These equations follow from the application of the first optimality condition to the Lagrangian equation (4.6).

$$\frac{\partial L}{\partial \boldsymbol{\lambda}_0} = (\mathbf{x}_0 - \mathbf{x}^0)^\mathsf{T} = \mathbf{0}^\mathsf{T} \tag{A.7}$$

$$\frac{\partial L}{\partial \boldsymbol{\lambda}_{i+1}} = \mathbf{g}_{i+1}^\mathsf{T}(\mathbf{u}_{i+1}, \mathbf{x}_i, \mathbf{x}_{i+1}, \boldsymbol{\theta}) = \mathbf{0}^\mathsf{T}, \quad \forall i = 0, 1, 2, ..., N_t - 1 \tag{A.8}$$

$$\frac{\partial L}{\partial \mu_{i+1}} = (\mathbf{y}_{i+1} - \mathbf{h}_{\boldsymbol{\theta},i+1}(\mathbf{u}_{i+1}, \mathbf{x}_{i+1}))^\mathsf{T} = \mathbf{0}^\mathsf{T}, \quad \forall i = 0, 1, 2, ..., N_t - 1 \tag{A.9}$$

$$\frac{\partial L}{\partial \mathbf{y}_{i+1}} = \frac{\partial J_{i+1}}{\partial \mathbf{y}_{i+1}} - \boldsymbol{\mu}_{i+1} = \mathbf{0}^\mathsf{T}, \quad \forall i = 0, 1, 2, ..., N_t - 1 \tag{A.10}$$

$$\frac{\partial L}{\partial \mathbf{x}_{N_t}} = \boldsymbol{\lambda}_{N_t}^\mathsf{T} \frac{\partial \mathbf{g}_{N_t}}{\partial \mathbf{x}_{N_t}} - \boldsymbol{\mu}_{N_t}^\mathsf{T} \frac{\partial \mathbf{h}_{\boldsymbol{\theta},N_t}}{\partial \mathbf{x}_{N_t}} = \mathbf{0}^\mathsf{T}, \tag{A.11}$$

$$\frac{\partial L}{\partial \mathbf{x}_i} = \boldsymbol{\lambda}_i^\mathsf{T} \frac{\partial \mathbf{g}_i}{\partial \mathbf{x}_i} + \boldsymbol{\lambda}_{i+1}^\mathsf{T} \frac{\partial \mathbf{g}_{i+1}}{\partial \mathbf{x}_i} - \boldsymbol{\mu}_i^\mathsf{T} \frac{\partial \mathbf{h}_{\boldsymbol{\theta},i}}{\partial \mathbf{x}_i} = \mathbf{0}^\mathsf{T}, \quad \forall i = 1, 2, ..., N_t - 1 \tag{A.12}$$

$$\frac{\partial L}{\partial \mathbf{x}_0} = \boldsymbol{\lambda}_1^\mathsf{T} \frac{\partial \mathbf{g}_1}{\partial \mathbf{x}_0} + \boldsymbol{\lambda}_0^\mathsf{T} = \mathbf{0}^\mathsf{T}, \tag{A.13}$$

$$\frac{\partial L}{\partial \mathbf{u}_{i+1}} = \frac{\partial J_{i+1}}{\partial \mathbf{u}_{i+1}} + \boldsymbol{\lambda}_{i+1}^\mathsf{T} \frac{\partial \mathbf{g}_{i+1}}{\partial \mathbf{u}_{i+1}} + \boldsymbol{\mu}_{i+1}^\mathsf{T} \frac{\partial \mathbf{h}_{\boldsymbol{\theta},i+1}}{\partial \mathbf{u}_{i+1}} = \mathbf{0}^\mathsf{T}, \quad \forall i = 0, 1, 2, ..., N_t - 1. \tag{A.14}$$

The procedures of the adjoint method for computing the gradient of the objective function (3.3), where the OPM-Flow simulator is used to solve the underlying reservoir dynamic state equations are summarized in the following Algorithm (3).

Algorithm 3: Computation of gradient of the objective function (3.3) using the adjoint method.

```
input: Prescribe the initial state $\mathbf{x}_0$ and input control $\mathbf{U} = \{\mathbf{u}_{i+1}\}_{i=0}^{N_t-1}$.
output: $\{\nabla_{\mathbf{u}_{i+1}} J\}_{i=0}^{N_t-1}$.
1  begin
2      forward simulation with OPM-Flow
3          Compute the state vector $\mathbf{x}_i$ and the corresponding output vector
           $\mathbf{y}_i$, $\forall i = 1, 2, ..., N_t$ using equations (2.19) and (2.21) respectively.
4      adjoint variable computation
5          for $i = 0, 1, ..., N_t - 1$ do
6              Compute $\boldsymbol{\mu}_{i+1} \in \boldsymbol{M}$ using Equations (A.10).
7          backward computation
8              for $i = N_t, N_t - 1, ..., 2, 1, 0$ do
9                  Compute $\boldsymbol{\lambda}_i \in \boldsymbol{\Lambda}$ using Equation (A.11) and (A.12)
10     Compute $\{\nabla_{\mathbf{u}_{i+1}} J\}_{i=0}^{N_t-1}$ using Equation (4.7).
```

A.3 Backpropagation method for gradient computation

To train the DNN model parameter $\boldsymbol{\Phi}$ in (5.7) for appropriate representation of the scalar-valued objective function (3.3), this study utilizes the following unregularized loss function (mean square error loss):

$$\mathcal{L}(\boldsymbol{\Phi}) = \frac{1}{2M} \sum_{\kappa \in \mathcal{T}} (h(\mathbf{U}_\kappa, \boldsymbol{\Phi}) - J(\mathbf{U}_\kappa))^2, \tag{A.15}$$

where $\{\mathbf{U}_\kappa, J(\mathbf{U}_\kappa)\}_{\kappa \in \mathcal{T}}$ is the given training data and cardinality of $\mathcal{T}$ is M. Here, the training process involves the minimization of (A.15). That's

$$\min_{\boldsymbol{\Phi}} \ \mathcal{L}(\boldsymbol{\Phi}). \tag{A.16}$$

This study utilizes Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) optimizer (Liu and Nocedal, 1989) to solve the unconstrained optimization problem (A.16). The L-BFGS method belongs to the family of quasi-Newton methods, which approximates the BFGS algorithm using a limited amount of computer memory. It is similar to a gradient descent algorithm with the search direction computed by preconditioning the gradient of $\mathcal{L}$ with curvature information.

The gradient of $\mathcal{L}$ with respect to each components of $\mathbf{\Phi}$ is computed using the Backpropagation algorithm explained as follows. Let's assume a ReLU activation function defined as follows:

$$\begin{aligned} g : \mathbb{R} &\longrightarrow [0,+\infty] \\ \gamma &\longmapsto g(\gamma) := \max(0,\gamma) = \begin{cases} 0, & \text{if } \gamma \leq 0 \\ \gamma, & \text{if } \gamma > 0 \end{cases} . \end{aligned} \tag{A.17}$$

Here, the derivative of g is computed as

$$g'(\gamma) := \begin{cases} 0, & \text{if } \gamma < 0 \\ 1, & \text{if } \gamma > 0 \end{cases} . \tag{A.18}$$

Mathematically, at $\gamma = 0$ the derivative of g is undefined, but by convention, most ML implementations take this derivative as zero or any value within the interval $[0,1]$.

For the DNN in Figure 5.4, the backpropagation method computes the gradient of the loss function with respective to the weights $\mathbf{W}^{(i)}$, $i = 1,2,3$ by chain rule, computing the gradient one layer at a time, iterating backward from the last layer. Given an arbitrary training example $(\mathbf{U}, J(\mathbf{U}))$, the loss function (A.15) becomes:

$$\mathcal{L}_1(\mathbf{\Phi}) = \frac{1}{2}(h(\mathbf{U},\mathbf{\Phi}) - J(\mathbf{U}))^2. \tag{A.19}$$

Using (5.3) - (5.7) and (A.17) - (A.18) and with assumption that $\gamma > 0$ in (A.18), the gradient of (A.19) is computed as follows.

Gradient of $\mathcal{L}_1(\mathbf{\Phi})$ with respect to the components of $\mathbf{W}^{(3)}$:

$$\begin{aligned} \frac{\partial \mathcal{L}_1}{\partial W^{(3)}_{N_3,j}} &= \frac{\partial \mathcal{L}_1}{\partial a^{(4)}_{N_3}} \frac{\partial a^{(4)}_{N_3}}{\partial W^{(3)}_{N_3,i}}, \qquad \forall i = 1,2,3 \\ &= \underbrace{(h(\mathbf{U},\mathbf{\Phi}) - J(\mathbf{U}))}_{\delta^{(L)} := \delta^{(4)}} a^{(3)}_i . \end{aligned} \tag{A.20}$$

Gradient of $\mathcal{L}_1(\mathbf{\Phi})$ with respect to the components of $\mathbf{W}^{(2)}$:

$$\begin{aligned}\frac{\partial \mathcal{L}_1}{\partial W_{i,j}^{(2)}} &= \frac{\partial \mathcal{L}_1}{\partial a_{N_3}^{(4)}} \frac{\partial a_{N_3}^{(4)}}{\partial a_i^{(3)}} \frac{\partial a_i^{(3)}}{\partial W_{i,j}^{(2)}}, \qquad \forall i = 1,2,..,N_2; j = 1,2,...,N_1 \\ &= \underbrace{(h(\mathbf{U},\mathbf{\Phi}) - J(\mathbf{U}))W_{N_3,i}^{(3)}}_{\delta_i^{(L-1)} := \delta_i^{(3)}} a_j^{(2)}. \end{aligned} \tag{A.21}$$

Gradient of $\mathcal{L}_1(\mathbf{\Phi})$ with respect to the components of $\mathbf{W}^{(1)}$:

$$\begin{aligned}\frac{\partial \mathcal{L}_1}{\partial W_{j,k}^{(1)}} &= \frac{\partial \mathcal{L}_1}{\partial a_{N_3}^{(4)}} \frac{\partial a_{N_3}^{(4)}}{\partial a_i^{(3)}} \frac{\partial a_i^{(3)}}{\partial a_j^{(2)}} \frac{\partial a_j^{(2)}}{\partial W_{j,k}^{(1)}}, \qquad \forall i = 1,2,..,N_2; j = 1,2,...,N_1; k = 1,2,...,N_u \\ &= \underbrace{(h(\mathbf{U},\mathbf{\Phi}) - J(\mathbf{U}))W_{N_3,i}^{(3)} W_{i,j}^{(2)}}_{\delta_{i,j}^{(L-2)} := \delta_{i,j}^{(2)}} u_k. \end{aligned} \tag{A.22}$$

The quantities $\delta_{i,j}^{(2)}, \delta_i^{(3)}$, and $\delta^{(4)}$ are usually called the propagation errors. For $M-$training examples, it is not hard to evaluate the gradient of $\mathcal{L}$ using (A.20) - (A.22).

Part II

Scientific contributions

Paper I

Robust Value Quantification of Enhanced Oil Recovery Methods using Ensemble-Based Optimization

Paper II

Ensemble-based constrained optimization using an exterior penalty method

Journal of Petroleum Science and Engineering 207 (2021) 109165

Contents lists available at ScienceDirect

Journal of Petroleum Science and Engineering

journal homepage: www.elsevier.com/locate/petrol

Ensemble-based constrained optimization using an exterior penalty method

Micheal B. Oguntola [a,b,*], Rolf J. Lorentzen [b]

[a] *University of Stavanger, 4036, Stavanger, Norway*
[b] *NORCE-Norwegian Research Center AS, 5838, Bergen, Norway*

A R T I C L E I N F O

Keywords:
Ensemble-based optimization
Stochastic gradient
Constrained optimization
Penalty method
Line search method
Production optimization
Lagrange method

A B S T R A C T

In science and engineering, non-linear constrained optimization has been a useful mathematical technique for many practical applications. Of interest to us is its applicability in the modeling and prediction of hydrocarbon reservoir production. In this paper, a new efficient, robust, and accurate optimal solution strategy based on the exterior penalty function (EPF) method and the adaptive ensemble-based optimization (EnOpt) approach (with backtracking line-search technique) for non-linear constrained optimization problems is presented. The purpose of this work is to provide a better user-friendly strategy which mitigates the problem often faced with the current constraints handling technique utilized when using the EnOpt method to solve constrained problems of water or EOR flooding. This study notes that the problem contributes to uncertainties in the gradient computation of the objective function and hence leads to the poor convergence rate of the standard EnOpt method. In this work, we used the EPF method to transform a given constrained optimization problem to a sequence of unconstrained subproblems and then sequentially solve the subproblems by unconstrained EnOpt procedure until convergence to the solution of the original problem. To demonstrate the advantage of the proposed methodology, we used it to solve analytical 2D bound constrained Rosenbrock's problem and a practical high dimensional bound constrained water flooding optimization problem associated with a 2D 5Spot field and a 3D Reek reservoir field. The numerical results are compared with EnOpt using classical Lagrangian approach, as well as the traditional EnOpt. Our findings showed that the proposed solution method has a fast convergence rate and is more accurate and robust.

1. Introduction

In reservoir production optimization, finding an injection/ production strategy (with high precision) for a particular oil recovery method that is economical at the expense of little or no negative environmental impact for a given reservoir type can be problematic. One reason for this problem is based on how the uncertain geological parameters in the reservoir of interest are quantified and utilized in the solution method for the reservoir optimization problem. Having a good quantification of the uncertain parameters using production history in reservoir studies has been the major contribution of closed-loop reservoir management (Aanonsen et al., 2009; Jansen et al., 2009; Jung et al., 2018; Zhang et al., 2019; Mirzaei-Paiaman et al., 2021). Several solution methods for the non-linear constrained optimization problems (Nocedal and Wright, 2006), usually encountered in hydrocarbon reservoir fields have been proposed and used extensively for different applications (see e.g., Sarma et al. (2005, 2006), Jansen (2011), Li and Reynolds (2011), Zhou et al. (2013), Xu et al. (2018) and Zhao et al. (2020)). For instance, of interest to us is the ensemble-based optimization (EnOpt) method, a popular and robust stochastic optimization technique, first introduced in Lorentzen et al. (2006), and further developed into its current form in Chen et al. (2009, 2010) and Fonseca et al. (2014). In EnOpt, the uncertainty descriptions in the reservoir are taken into account. However, the current constraint handling technique often utilized in the EnOpt method (see Chen et al. (2009)) poses uncertainty in the optimization result, which thus reduces its accuracy. In this study, we presented a more accurate means to deal with the constraints of the optimization problem in EnOpt by using the penalty function (PF) method (Nocedal and Wright, 2006; Rao, 2019). We demonstrate the convergence and added advantage (in terms of accuracy compared to the standard method) of the proposed coupling (of the PF method with EnOpt) using analytical and practical examples.

In constrained optimization problems, usually, one sorts to find the best feasible solution (out of a pull of solutions) called the optimal solution for the control variables that gives the extremum of a given objective function subject to a set of equality and/or inequality constraints. Here, feasibility implies that the underlying constraints are satisfied. In practice, the design of the control variables could take different forms. For instance, in reservoir optimization problems, the

* Corresponding author at: University of Stavanger, 4036, Stavanger, Norway.
E-mail address: micheal.b.oguntola@uis.no (M.B. Oguntola).

https://doi.org/10.1016/j.petrol.2021.109165
Received 17 February 2021; Received in revised form 22 June 2021; Accepted 25 June 2021
Available online 29 June 2021

control variables could include the number of well-type (producer or injector) to be sequentially drilled, their drilling order and location, the operational controls (such as the well-rates or bottom hole pressures over the production period), etc. Whence, the total number of control variables to be optimized is often of the order 100 and above. The objective function is usually some reservoir performance measures (such as net present value (NPV), oil recovery factor (ORF), etc.) defined on the given set of control variables. Several methods (with advantages and disadvantages Islam et al., 2020) have been established to find the optimal solutions to the optimization problems. They are majorly categorized as gradient-based and derivative-free techniques (Jesmani et al., 2020). Each of the techniques moves in the design solution space in a special pattern in search of the optimal solution. The gradient-based method mainly utilize (either analytically or by approximation) the derivatives of the objective function with respect to the control variables to control the search pattern while the derivative-free methods mostly use the objective function values in a stochastic way at each optimization iteration. The derivative-free methods are known to be computationally very efficient for low dimensional problems but struggle to converge once the dimension of problem is high (Arouri and Sayyafzadeh, 2020) (for a detailed review of the derivative-free methods, we refer interested readers to the work of Chen et al. (2020) and Semnani et al. (2021). Because of the large size of unknown variables often encounter in reservoir optimization problems, the gradient-based methods are more suitable to use.

Several gradient-based methods have been devised and utilized to solve different optimization problems such as well placement and control problems (Hutahaean et al., 2019; Liu et al., 2019; Sun et al., 2019; Epelle and Gerogiorgis, 2020). In terms of efficiency and accuracy, the adjoint method is ranked number one on the list of gradient-based methods because of its efficient and accurate gradient computation. However, for practical optimization applications such as the one in the management of subsurface hydrocarbon reservoirs, the adjoint method becomes very difficult to implement because (1) the adjoint-based gradient calculation with respect to the control variables is complicated and cannot be directly reused for different problems; (2) it requires user access to the reservoir simulator (Sarma et al., 2005, 2006; Jansen, 2011). To avoid the problem (1) faced with the adjoint method in well placement optimization, indirect approaches were developed (see, for e.g. Sarma et al. (2008) and Zandvliet et al. (2008)). However, problem (2) is inevitable. For these reasons, viable alternative solution methods that do not rely the adjoint method of gradient calculation are developed. The simplest one is the finite difference gradient approximation (FDGA) method which uses finite difference scheme to approximate each component of the gradient. In this case, at each iteration, the approximate gradient computation requires $2N_u$ function evaluations, where N_u is the size of the unknown variables to be optimized. Therefore, for high-dimensional problems, the FDGA method becomes computationally very expensive to use (Zhou et al., 2013; Jesmani et al., 2020). Other approximate gradient-based solution methods that are more efficient than the FDGA have been proposed in the literature. The most popular ones for reservoir optimization are the simultaneous perturbation stochastic approximation (SPSA) (Spall, 1998; Spall et al., 2006; Foroud et al., 2018), modified SPSA based on finite difference method (SPSA-FDG) (Zhou et al., 2013), the stochastic simplex approximation gradient (StoSAG) method Fonseca et al. (2017), and the ensemble-based optimization (EnOpt) method (Chen et al., 2010). These solution methods approximate the gradient of the objective function by simultaneously perturbing all the unknown variables at the same time, unlike the FDGA method where one variable is perturbed at a time. Theoretically, it has been proven by Do and Reynolds (2013) using the steepest descent scheme that a small difference in gradient computation exists between StoSAG, EnOpt, and SPSA (as well as its variants such as Gaussian-SPSA) methods. These methods have gained popularity recently due to their ability to incorporate uncertainty represented by multiple realizations of the reservoir model in their approximate gradient computation (Hutahaean et al., 2019; Jesmani et al., 2020).

The standard EnOpt is an iterative method, formulated based on a first-order inexact line search (with a simple backtracking technique Nocedal and Wright, 2006) steepest ascent optimization method (Snyman and Wilke, 2018). In this case, the approximate gradient of the objective function (in the line search direction) at each iterate is computed using a stochastic process. At each iteration, an ensemble of control vectors is sampled from a multivariate Gaussian distribution with a known mean (same as the current iterate) and a user-defined covariance matrix to compute the sample cross-covariance of the objective function and control variables. Using suitable assumptions, it is not hard to show that the sample cross-covariance is approximately the (regularized) gradient of the objective function (Chen et al., 2010; Do and Reynolds, 2013). Therefore, pressing issues with EnOpt will mainly be on the quantities (or inputs) that can impact the accuracy of the estimated gradient needed for a considerable increase in the objective function value at subsequent optimization iterations. In other words, uncertainty can be introduced by some quantities in the line search direction utilized in the EnOpt iterative method. Inappropriate choice of some of these quantities have proven to affect the convergence rate of the method. Sometimes, this could translate to the need for a higher number of iterations for convergence. To mention a few, quantities such as the ensemble size, sampling strategies, and distribution covariance matrix (or the perturbation size), etc., have been extensively studied in the literature. For instance, Fonseca et al. (2015) investigated the impact of ensemble size on the quality of the approximate EnOpt gradients (by comparing it with the exact adjoint gradient) for the Rosenbrock optimization problems and for a hydrocarbon reservoir. In their study, they provided a more computationally efficient and modified version of EnOpt using hypothesis testing. In other studies by Fonseca et al. (2014) and Stordal et al. (2016), they found that by systematically updating the perturbation size (through a method called covariance matrix adaptation (CMA)) at each optimization iteration would effectively improve the quality of the approximate gradient. Ramaswamy et al. (2020) evaluated the impact of different sampling strategies on the performance of the approximate EnOpt gradient for high-dimensional analytical and robust optimization problems. Their findings suggested the sampling design to consider in general for gradient approximation schemes in supersaturated cases, i.e., where the number of perturbation vector is less than the optimization unknowns. Zhang et al. (2018a,b) considered a slightly different iterative scheme, the trust region method with the conjugate gradient method rather than the line search method often utilized in EnOpt and demonstrated a fast convergence rate with applications on simple toy and synthetic reservoir problems.

Over recent years, the EnOpt method has received a lot of treatments towards improving its efficiency and accuracy, as mentioned above. However, in practice, we found that constrained optimization problems solved using the EnOpt technique were dealt with in an unconstrained manner. In reservoir optimization problems, there is usually a set of bound constraints imposed on the designed control variables. A common way to deal with this is to systematically discard or truncate values of control variables that do not satisfy the constraints, which leads to inaccurate gradient directions. An alternative is to use some transformation (see e.g., Chen et al. (2009) and Do and Reynolds (2013)) to enforce the constraints on the respective control variables. A transformation is not always advisable, especially for problems with complex non-linear constraints. Consequently, both truncation or transformation of control variables can contribute to uncertainties in the computation of the approximate gradient and hence lead to poor convergence to a desired local optimum. For this reason, we introduce a better and user-friendly approach, the PF method, to deal with constraints when using the EnOpt method to solve constrained optimization problems.

The PF method belongs to an important class of solution methods for constrained optimization problems. It transforms a given constrained problem into a sequence of unconstrained subproblems. See Nocedal and Wright (2006), Sun and Yuan (2006), Deb (2012) and Rao (2019) for a concrete theoretical overview of the PF method. In each subproblem, there exists a penalty function constructed by adding a penalty term (which is the constraint violation multiplied by a penalty parameter) to the objective function. Usually, the penalty term takes different forms depending on the type of constraint (equality or inequality). So, if an estimate of a control variable violates a given constraint, the objective function is penalized by an amount measured by the penalty term. More efficiently and robustly, the PF method works by sequentially solving each subproblem using a suitable unconstrained optimization procedure. In this case, the optimum found in one subproblem is utilized as the initial guess for the next unconstrained subproblem. In the subsequent subproblems, the solution improves gradually and eventually converges to the desired optimum of the original constrained optimization problem. However, we noted that the choice (in terms of initialization and subsequent adaptation) of the penalty parameter in the penalty term is very crucial to the convergence rate. Our study adopts the ideas in Nocedal and Wright (2006) and Rao (2019) for the adaption of the penalty parameter and hence the structure of the penalty term.

Over time, the PF method has evolved and its approach has been utilized to solve constrained problems in different areas of science and engineering. For instance, Zhang et al. (2016) used a variant of the PF method, the augmented Lagrangian method, and a stochastic gradient finite difference (SGFD) approach (Yan and Reynolds, 2014) to solve constrained oil reservoir optimization problem. They showed that the combined strategies give accurate results based on their comparison of SGFD and Gaussian distribution Simultaneous Perturbation Stochastic Approximation (G-SPSA) on simple high dimensional constrained analytical problem. However, the said SGFD is not very efficient compared to the EnOpt method because of the higher number of function evaluations and required storage associated with SGFD, especially for complex high dimensional constrained problems.

In this paper we present an efficient, accurate, and robust extension of EnOpt, to solve non-linear constrained optimization problems often encounter in science and engineering. We employ the exterior penalty function method with bracket operator (for inequality constraints) penalty term (Deb, 2012) to transform the original constrained optimization problems into a sequence of unconstrained subproblems and then utilize the adaptive EnOpt method as the unconstrained optimization procedure. For simplicity, we refer to the combined strategies as the EPF-EnOpt method. Our choice of the exterior PF is to allow for the flexibility of initialization for the unconstrained optimization procedure. We provide proof of convergence of the EPF method using suitable assumptions. Further, we demonstrate the use of the methodology with a challenging constrained analytical problem and practical constrained 2D 5-spot and 3D Reek oil reservoir problems and compare results with the standard EnOpt approach. In addition, we compare the EPF-EnOpt results with the classical Lagrangian constraint handling technique (similar to the formulation in Deb (2012) and Lu et al. (2017)) coupled with the standard EnOpt for the 3D Reek field to further illustrate the efficiency and accuracy of our proposed method. The Lagrange multiplier is estimated using a scheme similar to the one in Snyman and Wilke (2018). The rest of the paper is as follows; Section 2 discusses the mathematical model of constrained optimization problems. Section 3 looks at the theory of the exterior PF formulation for general constrained optimization problems and the derivation of the adaptive EnOpt procedures. Section 4 builds on previous sections to formulate the EPF-EnOpt algorithm and discuss its convergence, and finally, Sections 5 and 6 present applications (with relevant discussion) and conclusion, respectively.

2. Constrained optimization problem and techniques

The mathematical model of constrained optimization is useful for many applications in science and engineering. For instance, in hydrocarbon reservoir management, resources such as the injecting and producing facilities are limited in capacities. Therefore, the optimization of the objective function, usually a given reservoir performance index (such as the oil recovery factor or the net present value (NPV), etc.) is necessarily subject to a well-defined set of constraints on the designed control variables (such as the water rate of each injecting well at each control time step, etc.). This is usually referred to as an optimal control problem during reservoir development, and the goal of this problem is to find the best (optimal) strategy of control variables for maximum profit.

In this section, we present a general constrained optimization (maximization) problem, often encountered in science and engineering. Let $\mathbf{u} \in \mathbb{R}^{N_u}$ be the vector of designed control variables (optimization unknowns) i.e, $\mathbf{u} = [u_1, u_2, \dots, u_{N_u}]^\top$ (⊤ means transpose). Again, the form of $\mathbf{u}$ can differ for different problems. In reservoir optimization problems, the components of $\mathbf{u}$ can represent the wells (injectors or producers) target rates or bottom hole pressures in a specific control time step during water flooding. The general N_u−dimensional constrained optimization problem is to find the optimum /best $\mathbf{u} \in \mathbb{R}^{N_u}$ that

$$\underset{\mathbf{u}\in\mathbb{R}^{N_u}}{\text{maximize}} \quad J(\mathbf{u}) \tag{1}$$

$$\text{subject to:} \quad g_i(\mathbf{u}) \geq 0, \quad \forall i \in \text{I} \tag{2}$$

$$h_j(\mathbf{u}) = 0, \quad \forall j \in \text{E}, \tag{3}$$

where J is the objective function (from $\mathbb{R}^{N_u}$ into $\mathbb{R}$), g_i and h_j are the underlying constraint functions (from $\mathbb{R}^{N_u}$ into $\mathbb{R}$ respectively), I and E are the indexing sets for the inequality and equality constraints respectively. The optimization problem stated in Equations (1) - (3) becomes unconstrained should $\text{I} \cup \text{E} = \emptyset$. Since,

$$\max_{\mathbf{u}\in\mathbb{R}^{N_u}} \quad J(\mathbf{u}) = -\min_{\mathbf{u}\in\mathbb{R}^{N_u}} (-J(\mathbf{u})), \tag{4}$$

without the loss of generality, the maximization problem (1) - (3) can be considered as a minimization problem by replacing (1) with (4). Therefore, the focus of this study is on constrained minimization problems of the type:

$$\min_{\mathbf{u}\in\mathbb{R}^{N_u}} \quad f(\mathbf{u}) \tag{5}$$

$$\text{subject to:} \quad g_i(\mathbf{u}) \geq 0, \quad \forall i \in \text{I} \tag{6}$$

$$h_j(\mathbf{u}) = 0, \quad \forall j \in \text{E}, \tag{7}$$

where f is a continuous objective function. The structure of the constraints in Eqs. (6)–(7) varies from one problem to another. In reservoir optimization problems constraints are often given as linear inequality constraints. The simplest type are bound constraints where each inequality only depends on a single control variable. In this case, suppose that u_i^{low} and u_i^{upp} are the lower and upper bounds respectively for each control variable u_i, the bound constraints are given as:

$$u_i^{\text{low}} \leq u_i \leq u_i^{\text{upp}}, \qquad \forall i \in 1, 2, \dots, N_u. \tag{8}$$

Constraint types called the "output constraints" in the petroleum industry are also very commonly utilized during petroleum production. These are non-linear constraints that represent operational limits and they are usually evaluated using the reservoir simulator. A typical example is pressure limits for injectors or producers in a model where the wells are operated using flow rate targets (given by the control variables). Our primary goal with this paper is handling of bound constraints, but, it is not limited to this alone as the exterior PF methodology can be used for complicated constraints (like the output constraints). However, as a common practice, one could let the reservoir simulator handle the output constraints. Next, we explain two traditional ways in the literature to handle the bound constraints, and then we continue with the methodology using the exterior PF.

2.1. Bound constraints transformation

Suppose that Eq. (8) is the only set of constraints impose on the control variables in the optimization problem (5)–(7). A given optimal solution method like the EnOpt method finds an approximate solution that solves the optimization problem until convergences. At each optimization iteration, to ensure that each solution obtained is feasible, i.e., the underlying constraints are satisfied, either of the following two procedures is useful.

1. Linear transformation with truncation.
   For each $i \in \{1, 2, \dots, N_u\}$, a bijective linear function is defined to transform the domain of the control variable u_i into the closed interval $[0, 1]$. That is

$$T_i : [u_i^{\text{low}}, u_i^{\text{upp}}] \longrightarrow [0, 1], \tag{9}$$
$$u_i \longmapsto T_i(u_i) := \hat{u}_i = \frac{u_i - u_i^{\text{low}}}{u_i^{\text{upp}} - u_i^{\text{low}}}, \qquad u_i^{\text{upp}} \neq u_i^{\text{low}}$$

   (this transformation follows from the result of a simple algebraic rearrangement of the inequalities in Eqs. (8), and then divide through by $u_i^{\text{upp}} - u_i^{\text{low}}$), where $\hat{u}_i$ is the transformed control variable u_i. In this case, the optimization process is carried out in the interval $[0, 1]$. Because the linear function (9) is bijective, its inverse exist. Thus, any value of $\hat{u}_i$ found can easily be transformed into its equivalent value in the domain $[u_i^{\text{low}}, u_i^{\text{upp}}]$ by using;

$$u_i = (u_i^{\text{upp}} - u_i^{\text{low}})\hat{u}_i + u_i^{\text{low}}. \tag{10}$$

   Here, any value of $\hat{u}_i$ that falls outside the closed interval $[0, 1]$ is systematically approximated (or truncated) as follows;

$$\hat{u}_i = \begin{cases} 1, & \text{if } \hat{u}_i > 1 \\ 0, & \text{if } \hat{u}_i < 0. \end{cases} \tag{11}$$

   Vital estimates of control variables for accurate gradient directions can easily be lost using this truncation. As a consequence, this can affect the convergence rate of the solution method.
2. Logarithmic transformation.
   Here, for each u_i, $i \in \{1, 2, \dots, N_u\}$, a logarithmic (log) function is used to transform the domain (excluding the boundary points) of u_i to the entire set of real numbers i.e, $\mathbb{R} = (-\infty, +\infty)$. The transformation is defined by:

$$L_i : (u_i^{\text{low}}, u_i^{\text{upp}}) \longrightarrow (-\infty, +\infty) \tag{12}$$
$$u_i \longmapsto L_i(u_i) := \hat{u}_i = \log_e\Big(\frac{u_i - u_i^{\text{low}}}{u_i^{\text{upp}} - u_i}\Big).$$

   In this case, the optimization procedure occurs in the transformed domain, $(-\infty, +\infty)$. Therefore, since $[0, 1] \subset (-\infty, +\infty)$, the transformed optimization unknown, $\hat{u}_i$ can now fluctuate or vary in a set of points larger than the optimization domain obtained from the linear transformation. Similarly, L_i is a well-defined bijective function and hence its inverse can be computed as;

$$u_i = \frac{u_i^{\text{upp}} \exp(\hat{u}_i) + u_i^{\text{low}}}{\exp(\hat{u}_i) + 1}. \tag{13}$$

   Using the log transformation helps to reduce the possibility of truncation often encounter with the linear transformation. However, if the initial solution guess is close to the boundary, it is difficult to find a suitable gradient direction for improvements. In addition, problems in which the optimal solution of one or more control variables lie on the boundary of the solution domain can be hard to solve. It is because of the following simple observation (from Eq. (12));

$$\hat{u}_i = \begin{cases} +\infty, & \text{if } u_i = u_i^{\text{low}} \\ -\infty, & \text{if } u_i = u_i^{\text{upp}} \end{cases} \tag{14}$$

   whereas, neither $u_i^{\text{low}}, u_i^{\text{upp}} \in (u_i^{\text{low}}, u_i^{\text{upp}})$ nor $-\infty, +\infty \in (-\infty, +\infty)$.

Application of any of the described procedures above transforms the given constrained optimization problem (with only bound constraints) in Eqs. (5)–(7) to an unconstrained optimization problem. The resulting unconstrained optimization problem is then solved by suitable unconstrained minimization methods (like the ones considered in Li and Reynolds (2011), Do and Reynolds (2013), Zhao et al. (2013) and Zhou et al. (2013), etc.) and thus leading to the solution of the initial/original constrained optimization problem (5)–(7). This has been a common practice, especially in the petroleum industries.

In addition to the shortcomings mentioned above, the procedures of handling bound constraints cannot easily be extended to more complicated (possibly non-linear) equality or inequality constraints. Next in this study, we present a more accurate method to solve a general constrained minimization problems.

3. Exterior PF formulation

Suppose that a given constrained optimization problem is in the form of Eqs. (5)–(7). Let $\mathcal{D} \subset \mathbb{R}^{N_u}$ be the domain of feasible solutions. Hence, $\mathbb{R}^{N_u} \backslash \mathcal{D}$ is the set of infeasible points. To solve the problem, first, we transform it into a sequence of unconstrained subproblems, $\{P_k\}_{k=1}^{\infty}$ using the exterior quadratic PF method. In each subproblem, P_k, $k = 1, 2, \dots$, is a penalty function defined as follows;

$$P_k : \mathbb{R}^{N_u} \times (0, +\infty) \longrightarrow \mathbb{R} \tag{15}$$
$$(\mathbf{u}, r_k) \longmapsto P_k(\mathbf{u}, r_k) = f(\mathbf{u}) + r_k \underbrace{\Big(\sum_{i \in \mathrm{I}} (\min\{g_i(\mathbf{u}), 0\})^2 + \sum_{j \in \mathrm{E}} |h_j(\mathbf{u})|^2\Big)}_{Q(\mathbf{u})}$$

(we set, $P_k := P_k(\mathbf{u}, r_k)$), where $\{r_k\}_{k=1}^{\infty}$ is an increasing sequence of positive penalty parameters (which control the iteration of P_k) well-defined such that

$$\lim_{k \to +\infty} r_k = +\infty. \tag{16}$$

For convenience, we consider a simple sequence of penalty parameters given by the relation

$$r_{k+1} = c r_k, \quad k = 1, 2, \dots, \tag{17}$$

where $c \geq 1$ and the first term, $r_1 > 0$ are carefully selected constants. Different values of r_1 can mean different number of subproblems to solve before convergence. Cases of subproblems with exact penalty functions have been treated extensively (we refer readers to Han and Mangasarian (1979) and Nocedal and Wright (2006) for more information on the impact of the different selection criteria for r_1). For each $i \in \mathrm{I}$, we define the bracket operator on $g_i(\mathbf{u})$ as

$$\min\{g_i(\mathbf{u}), 0\} = \begin{cases} g_i(\mathbf{u}), & \text{if } g_i(\mathbf{u}) < 0 \\ & \text{(i.e. } \mathbf{u} \in \mathbb{R}^{N_u} \backslash \mathcal{D} \text{ (constraint is violated))} \\ 0, & \text{if } g_i(\mathbf{u}) \geq 0 \\ & \text{(i.e. } \mathbf{u} \in \mathcal{D} \text{ (constraint is satisfied))} \end{cases} \tag{18}$$

and for each $j \in \mathrm{E}$,

$$|h_j(\mathbf{u})| = \begin{cases} -h_j(\mathbf{u}), & \text{if } h_j(\mathbf{u}) < 0 \\ & \text{(i.e. } \mathbf{u} \in \mathbb{R}^{N_u} \backslash \mathcal{D} \text{ (constraint is violated))} \\ 0, & \text{if } h_j(\mathbf{u}) = 0 \\ & \text{(i.e. } \mathbf{u} \in \mathcal{D} \text{ (constraint is satisfied))} \\ h_j(\mathbf{u}), & \text{if } h_j(\mathbf{u}) > 0 \\ & \text{(i.e. } \mathbf{u} \in \mathbb{R}^{N_u} \backslash \mathcal{D} \text{ (constraint is violated)).} \end{cases} \tag{19}$$

Since h_j is a real-valued function, then $|h_j|^2 = h_j^2$, and hence Eq. (19) can be neglected. However, in a situation where an l_1-penalty function definition (Nocedal and Wright, 2006) is used in Eq. (15) or a complex-valued function h_j is considered, then (19) is retained. In Eq. (15), Q is called the exterior penalty term defined based on the given constraints (6)–(7). When a given estimate of the control vector $\mathbf{u}$ is infeasible to a given constraint, it means a violation of the constraint by the estimate. For such a violation, we penalize the objective function by an amount measured by Q. Therefore, using Eqs. (18) and (19), it is not hard to see that for a feasible solution, Q is zero and for an infeasible solution, Q is positive with an amount proportional to the square of the value given by Eq. (18) (for inequality constraint) and/ or Eq. (19) (for equality constraints). So, in general:

$$Q(\mathbf{u}) \geq 0, \quad \forall \mathbf{u} \in \mathbb{R}^{N_u}. \tag{20}$$

Next, we sequentially (with successively increasing r_k values) solve the subproblem

$$\min_{\mathbf{u}\in\mathbb{R}^{N_u}} \; P_k(\mathbf{u}, r_k), \quad \forall k = 1, 2, \ldots, \tag{21}$$

where P_k is defined by Eq. (15) (and we will often replace the pair $(\mathbf{u}, r_k)$ by $\mathbf{u}(r_k)$ in the rest of this section) and r_k is defined by Eq. (17). This is an unconstrained optimization problem. Note that P_k is non-differentiable at the points $\mathbf{u}$ that lie on the border between the feasible and the infeasible domains (because of Eqs. (18) and (19)), that is its gradient does not exist at the said points. Therefore, unconstrained numerical solution methods based on analytic gradient computation is not suitable. The bundle methods solve non-differentiable optimization problems more effectively and are reliable (Bagirov et al., 2014). However, for an extensive application, we instead utilize the EnOpt method, which does not rely directly on analytic gradient. For any $r_k, k = 1, 2, \ldots$, the EnOpt method finds an approximate minimum point of P_k, denoted by $\mathbf{u}_k^* := \mathbf{u}^*(r_k)$. Usually, P_k possesses a minimum as a function of $\mathbf{u}$ in the infeasible region, especially when its initial guess, $\mathbf{u}_{k-1}$ is an infeasible point. The sequence of unconstrained minima denoted by $\mathbf{u}_k^*, k = 1, 2, \ldots$ converges to the desired minimizer of the original constrained optimization problem (5)–(7) as $k \to +\infty$. Therefore, $\mathbf{u}_k^*$ approaches the feasible domain gradually, and eventually lies in the feasible region (equivalently, $|Q(\mathbf{u}_k^*)| \to 0$) as $r_k \to +\infty$, with $k \to +\infty$.

The EnOpt method for solving the unconstrained optimization problem of the type presented in Eq. (21) is next described.

3.1. EnOpt procedures

We consider the problem (21) for any fixed penalty parameter, r_k. The EnOpt is an iterative optimal solution method in which the user starts by selecting an initial control vector $\mathbf{u}_1$ according to some rules, and then proceed to find the best approximation to the optimum solution $\mathbf{u}$, that minimizes $P_k(\mathbf{u}, r_k)$ using a preconditioned gradient descent method given by

$$\mathbf{u}_{l+1} = \mathbf{u}_l - \frac{1}{\beta_l}\frac{\mathbf{C}_\mathbf{u}^l \mathbf{G}_l^\top}{\|\mathbf{C}_\mathbf{u}^l \mathbf{G}_l^\top\|_2}, \quad \forall l = 1, 2, \ldots, \tag{22}$$

until convergence, where l is the iteration index; $\mathbf{G}_l$ is the sensitivity (an approximate gradient) of $P_k(\mathbf{u}, r_k)$ with respect to the control variables $u_i, i = 1, 2, \ldots, N_u$, also called the search direction at the lth iteration; β_l is the tuning parameter for iteration step size. It is used to ensure a sufficient descend along the search direction. Here, we employ the backtracking line search method (Nocedal and Wright, 2006) to compute (and update when necessary) β_l at each iteration (see Algorithm 1). $\mathbf{C}_\mathbf{u}^l$ denotes a real symmetric positive definite matrix defined as the covariance matrix of control variables at the lth iteration; $\|.\|_2$ is the l_2- norm. In hydrocarbon reservoir management, it is not very common to have controls (like water rate and oil rate) at different wells (injector and producer) to correlate. However, the water rate (for example) of a particular injection well can be correlated in time throughout the production period in a water flooding scenario. Because of this, in Eq. (22), $\mathbf{C}_\mathbf{u}^l$ is defined to avoid inappropriate dependence among control variables. It also ensures possible smoothness, correlation, and long-term fluctuations in the control variables of the same kind (Do and Reynolds, 2013).

We initialize the covariance matrix, $\mathbf{C}_\mathbf{u}^l$ differently at the start of iteration, $l = 1$ for the different optimization problems considered in this study. For analytical problem (like the 2D Rosenbrock problem Snyman and Wilke, 2018) where the N_u-unknowns are distinct and do not need to be correlated over time, we used a $N_u \times N_u-$ diagonal matrix where the variances of unknown variables are the diagonal elements. Suppose that $\sigma_i^2, i = 1, 2, 3, \ldots, N_u$ is the variance of the unknown control variable $u_i, i = 1, 2, \ldots, N_u$ respectively. The initial covariance matrix is taken as

$$\mathbf{C}_\mathbf{u}^1 = \begin{pmatrix} \sigma_1^2 & 0 & 0 & \ldots & 0 \\ 0 & \sigma_2^2 & 0 & \ldots & 0 \\ . & & & & \\ . & & & & \\ . & & & & \\ 0 & 0 & 0 & \ldots & \sigma_{N_u}^2 \end{pmatrix}. \tag{23}$$

For the water flooding reservoir optimization problem considered in this study, we initialize the covariance matrix, $\mathbf{C}_\mathbf{u}^l$ using a stationary AR(1) model (similar to the one in Oguntola and Lorentzen (2020)) to simulate the correlation of control variables at individual well and assume control variables at different wells are not correlated. To achieve this scenario, we used the following covariance function:

$$\mathrm{Cov}(u^m[t], u^m[t+h]) = \sigma_m^2 \rho^h \left(\frac{1}{1-\rho^2}\right), \quad \forall\, h \in [0, N_t - t], \tag{24}$$

where $u^m[t]$ is the control variable of well $m = 1, 2, \ldots, N_{\text{well}}$ at the control time step t, N_{well} is the total number of wells, σ_m^2 is the variance for the well m, and ρ is the correlation coefficient used to introduce some dependence between controls of individual wells at different control time steps. Since the AR(1) model is stationary, then $\rho \in (-1, 1)$. This formulation (in addition to using the symmetric property of covariance matrix) gives a block diagonal matrix, $\mathbf{C}_\mathbf{u}^1$ of the form

$$\mathbf{C}_\mathbf{u}^1 = \begin{pmatrix} \mathbf{C}_{\mathbf{u}^1}^1 & 0 & 0 & \ldots & 0 \\ 0 & \mathbf{C}_{\mathbf{u}^2}^1 & 0 & \ldots & 0 \\ . & & & & \\ . & & & & \\ . & & & & \\ 0 & 0 & 0 & \ldots & \mathbf{C}_{\mathbf{u}^{N_{\text{well}}}}^1 \end{pmatrix}. \tag{25}$$

In Eq. (25), $\mathbf{C}_{\mathbf{u}^m}^1$ is the user-defined covariance matrix (obtained by using Eq. (24)) to ensure some degree of correlation and smoothness on the control vector, $\mathbf{u}^m$ for the well $m = 1, 2, \ldots, N_{\text{well}}$ at iteration $l = 1$; $\mathbf{u}^m = [u_1^m, u_2^m, \ldots, u_{N_t}^m]$ is the vector of control variables at well m for all the N_t control time steps. For example, u_i^m denotes the control variable for well m at the ith control time step. In general, for this problem, the complete optimization unknowns (control vector) can be written as;

$$\begin{aligned} \mathbf{u} &= [\mathbf{u}^1, \mathbf{u}^2, \ldots, \mathbf{u}^{N_{\text{well}}}]^\top = [\{u_n^m\}_{n=1}^{N_t} \,:\, m = 1, 2, \ldots, N_{\text{well}}]^\top \\ &= [u_1, u_2, \ldots, u_{N_u}]^\top \end{aligned} \tag{26}$$

where $N_u = N_{\text{well}} \times N_t$ is the total number of control variables.

At subsequent iterations, $l \neq 1$, the covariance matrices in (23) and (25) are updated using the statistical approach presented in Stordal et al. (2016) to get an improved covariance matrix, $\mathbf{C}_\mathbf{u}^{l+1}$. In the EnOpt community, this process is called the Covariance Matrix Adaptation (CMA)-EnOpt method (Fonseca et al., 2014) (we shall refer to this as simply the "standard EnOpt method"). Also, in Eq. (22), pre-multiplying $\mathbf{G}_l^\top$ by $\mathbf{C}_\mathbf{u}^l$ has shown to produce a better performance, see e.g. Amari (1998). Indeed, the product $\mathbf{C}_\mathbf{u}^l \mathbf{G}_l^T$ is the natural gradient for this problem. It is independent of the problem parameterization and accounts for gradient uncertainty.

The preconditioned approximate gradient, $\mathbf{C}_\mathbf{u}^l \mathbf{G}_l^\top$ is computed as follows; At the lth iteration, we sample N control vectors, $\mathbf{u}_{l,j}$, $j =$

$1, 2, \dots, N$ from a multivariate normal distribution with mean equal to the lth control vector, $\mathbf{u}_l$, and covariance matrix $\mathbf{C}^l_{\mathbf{u}}$, i.e., $\mathbf{u}_{l,j} \sim \mathcal{N}(\mathbf{u}_l, \mathbf{C}^l_{\mathbf{u}})$, $j = 1, 2, \dots, N$, where N is the number of perturbations. Here, the subscript j is used to identify the perturbation vector, $\mathbf{u}_{l,j}$ and hence separate it from the one obtained by optimization iteration (see Eq. (22)). Each perturbation vector, $\mathbf{u}_{l,j}, j = 1, 2, \dots, N$ is then coupled with the penalty parameter r_k to compute the penalty function values, $P_k(\mathbf{u}_{l,j}, r_k), j = 1, 2, \dots, N$ (see Eq. (15)). Next, we utilize the perturbations to compute the approximate sample cross-covariance of the control vector $\mathbf{u}_l$ and the objective function $P_k(u_l, r_k)$ as follows (Fonseca et al., 2017):

$$\mathbf{C}^l_{\mathbf{u},P_k(\mathbf{u},r_k)} \approx \frac{1}{N-1}\sum_{j=1}^{N}(\mathbf{u}_{l,j} - \mathbf{u}_l)\Big(P_k(\mathbf{u}_{l,j}, r_k) - P_k(\mathbf{u}_l, r_k)\Big). \tag{27}$$

Note that, in this case, we have used the fact that the mean of $\{\mathbf{u}_{l,j}\}_{j=1}^N$ can be approximated by $\mathbf{u}_l$, since

$$\mathbf{u}_{l,j} \sim \mathcal{N}(\mathbf{u}_l, \mathbf{C}^l_{\mathbf{u}}),\ \ j = 1, 2, \dots, N.$$

By first order Taylor series expansion of $P_k(\mathbf{u}, r_k)$ about $\mathbf{u}_l$, one can easily show that Eq. (27) is an approximation of $\mathbf{C}^l_{\mathbf{u}}\mathbf{G}^{\top}_l$. Since r_k is fixed, then it is not hard to see (by Taylor expansion about $\mathbf{u}_l$) that,

$$\begin{aligned} P_k(\mathbf{u}, r_k) &= P_k(\mathbf{u}_l, r_k) + \left[\frac{\partial P_k(\mathbf{u}_l, r_k)}{\partial \mathbf{u}}\right]^{\top}(\mathbf{u} - \mathbf{u}_l) + O(\|(\mathbf{u} - \mathbf{u}_l)\|^2) \\ \implies P_k(\mathbf{u}, r_k) - P_k(\mathbf{u}_l, r_k) &= \left[\frac{\partial P_k(\mathbf{u}_l, r_k)}{\partial \mathbf{u}}\right]^{\top}(\mathbf{u} - \mathbf{u}_l) + O(\|(\mathbf{u} - \mathbf{u}_l)\|^2). \end{aligned} \tag{28}$$

Pre-multiply both sides of (28) by $(\mathbf{u} - \mathbf{u}_l)$ and set $\mathbf{u} = \mathbf{u}_{l,j}$, we get

$$\begin{aligned} &(\mathbf{u}_{l,j} - \mathbf{u}_l)(P_k(\mathbf{u}_{l,j}, r_k) - P_k(\mathbf{u}_l, r_k)) \\ &\quad = (\mathbf{u}_{l,j} - \mathbf{u}_l)\mathbf{G}^{\top}_l(\mathbf{u}_{l,j} - \mathbf{u}_l) + O(\|(\mathbf{u}_{l,j} - \mathbf{u}_l)\|^3), \quad \forall j = 1, 2, \dots, N, \end{aligned} \tag{29}$$

where, $\mathbf{G}^{\top}_l = \frac{\partial P_k(\mathbf{u}_l, r_k)}{\partial \mathbf{u}}$ is the value of the approximate gradient of P_k at $(\mathbf{u}_l, r_k)$ and $O(\|(\mathbf{u}_{l,j} - \mathbf{u}_l)\|^3)$ is the remaining terms containing higher order (≥ 3) of $(\mathbf{u}_{l,j} - \mathbf{u}_l)$. Assuming that the magnitude of the difference, $(\mathbf{u}_{l,j} - \mathbf{u}_l)$ is very small, then with first order Taylor expansion $\big($ by neglecting, $O(\|(\mathbf{u}_{l,j} - \mathbf{u}_l)\|^3)$ $\big)$, we obtain the following from (29).

$$\begin{aligned} &\frac{1}{N-1}\sum_{j=1}^{N}(\mathbf{u}_{l,j} - \mathbf{u}_l)(P_k(\mathbf{u}_{l,j}, r_k) - P_k(\mathbf{u}_l, r_k)) \\ &\qquad\qquad \approx \Big(\frac{1}{N-1}\sum_{j=1}^{N}(\mathbf{u}_{l,j} - \mathbf{u}_l)(\mathbf{u}_{l,j} - \mathbf{u}_l)\Big)G^{\top}_l \\ &\implies \mathbf{C}^l_{\mathbf{u},P_k(\mathbf{u},r_k)} \approx \mathbf{C}^l_{\mathbf{u}}\mathbf{G}^{\top}_l. \end{aligned} \tag{30}$$

The iterative scheme of the EnOpt (see Eq. (22)) is repeated until a specified stopping (or convergence) criteria is satisfied. In this study, we used the following criteria

$$\frac{|P_k(\mathbf{u}_{l+1}, r_k) - P_k(\mathbf{u}_l, r_k)|}{|P_k(\mathbf{u}_l, r_k)|} < \epsilon_3, \tag{31}$$

where ϵ_3 is a specified tolerance.

3.1.1. Backtracking line search method

The backtracking method (similar to the one in Nocedal and Wright (2006)) considered in this study is demonstrated in Algorithm 1 (the Armijo condition for sufficient decrease).

Algorithm 1: Procedure for step size selection

```
Step 1. Fix parameters $\alpha_1 \in (0,1)$ and $\alpha_2 \in (0,1)$.
Step 2. Start iteration with step size $\lambda := \frac{1}{\beta_1} > 0$.
while $P(\mathbf{u}_{l+1}, r_k) \geq P(\mathbf{u}_l, r_k) - \alpha_2\lambda\nabla^{\top}_{\mathbf{u}}P(\mathbf{u}_l, r_k)\mathbf{g}_l$ do
    $\lambda = \alpha_1\lambda$
end while
```

In Algorithm 1, $\mathbf{g}_l$ is the search direction at the lth iteration evaluated using,

$$\mathbf{g}_l = \frac{\nabla^{\top}_{\mathbf{u}}P(\mathbf{u}_l, r_k)}{\|\nabla^{\top}_{\mathbf{u}}P(\mathbf{u}_l, r_k)\|_2},$$

$\nabla^{\top}_{\mathbf{u}}P(\mathbf{u}_l, r_k)$ is the regularized approximate gradient of $P(\mathbf{u}_l, r_k)$ computed using Eq. (27), α_1 is the backstepping or step size contraction parameter, and α_2 is a given constant. From vector dot product, it is clear that,

$$\begin{aligned} \nabla^{\top}_{\mathbf{u}}P(\mathbf{u}_l, r_k)\mathbf{g}_l &= \|\nabla^{\top}_{\mathbf{u}}P(\mathbf{u}_l, r_k)\|_2\|\mathbf{g}_l\|_2\cos\theta \\ &= \|\nabla^{\top}_{\mathbf{u}}P(\mathbf{u}_l, r_k)\|_2\left\|\frac{\nabla^{\top}_{\mathbf{u}}P(\mathbf{u}_l, r_k)}{\nabla^{\top}_{\mathbf{u}}P(\mathbf{u}_l, r_k)}\right\|_2\cos\theta \\ \implies \nabla^{\top}_{\mathbf{u}}P(\mathbf{u}_l, r_k)\mathbf{g}_l &= \|\nabla^{\top}_{\mathbf{u}}P(\mathbf{u}_l, r_k)\|_2, \quad (\text{since } \theta = 0) \end{aligned} \tag{32}$$

where θ is the angle between the vectors $\nabla^{\top}_{\mathbf{u}}P(\mathbf{u}_l, r_k)$ and $\mathbf{g}_l$. Therefore, for convenience, we utilize Eq. (32) in Algorithm 1.

4. EPF-EnOpt method

The combined strategies of the Exterior PF method with the EnOpt method give rise to a more accurate and robust optimal solution method, the EPF-EnOpt method, for constrained optimization problems. More concisely, we present the workflow of the EPF-EnOpt procedures in Algorithm 2.

Algorithm 2: Procedures for solving constrained optimization problems

```
Step 1. Given tolerances $\epsilon_1 > 0$ and $\epsilon_2 > 0$, a suitable initial penalty parameter $r_1 > 0$, and a growth constant $c \geq 1$, initial starting point (feasible or infeasible) $\mathbf{u}_1 := \mathbf{u}(r_1) \in \mathbb{R}^{N_u}$. Set $k = 1$.
Step 2. Formulate the term, $P_k$ of the sequence $\{P_k\}_{k=1}^{\infty}$, using Eq. (15).
Step 3. Find a solution $\mathbf{u}^*_k := \mathbf{u}^*(r_k)$ of the unconstrained minimization problem stated in Eq. (21) using the EnOpt procedures (in sub-section 3.1).
Step 4. Check the stopping criteria:
if $|P_k(\mathbf{u}^*(r_k)) - P_k(\mathbf{u}(r_k))| < \epsilon_1$ then
    if $|Q(\mathbf{u}^*(r_k))| \leq \epsilon_2$ then
        set, $\mathbf{u}^* = \mathbf{u}^*_k$ $\big(\mathbf{u}^*$ is the solution of the original problem (5)–(7)$\big)$
and terminate;
    else
        go to Step 5
    end if
else
    go to Step 5
end if
Step 5. Select $r_{k+1} = cr_k, \mathbf{u}_{k+1} = \mathbf{u}^*_k$. Set $k = k+1$, and turn to Step 2.
```

4.1. Convergence of the EPF method

In this section, we consider the problem (5)–(7) where f, $g_i, \forall i \in \mathrm{I}$, and $h_j, \forall j \in \mathrm{E}$ are continuous functions defined on $\mathbb{R}^{N_u}$ and a sequence of penalty parameters, $\{r_k\}_{k=1}^{\infty}$ that satisfies the condition of Eq. (16). Assume that for each $k = 1, 2, \dots$, $\mathbf{u}^*_k := \mathbf{u}^*(r_k)$ is the optimal solution (or the desired minimizer) of the unconstrained optimization problem (21). Also, we assume that the desired optimal solution, $\mathbf{u}^*$ of the original constrained optimization problem (5)–(7) exist and is unique. We establish in Lemma 4.1, the basic properties of the exterior PF formulation, and the proof of convergence of the EPF procedures is thereafter presented.

Lemma 4.1. *Suppose that $\{r_k\}_{k=1}^{\infty}$ is a strictly increasing sequence, i.e. $0 < r_k < r_{k+1}, \forall k = 1, 2, \dots,$ then*

1. $P_k(\mathbf{u}_k^*) \le P_{k+1}(\mathbf{u}_{k+1}^*)$,
2. $f(\mathbf{u}_k^*) \le f(\mathbf{u}_{k+1}^*)$,
3. $Q(\mathbf{u}_k^*) \ge Q(\mathbf{u}_{k+1}^*)$.

Proof.

1. Using the definition of $P_k, k = 1, 2, \ldots$, in Eq. (15) and since $\mathbf{u}_k^*$ and $\mathbf{u}_{k+1}^*$ are the minimizers of P_k and P_{k+1} respectively, we have

$$P_k(\mathbf{u}_k^*) \le P_k(\mathbf{u}_{k+1}^*), \tag{33}$$

and

$$P_{k+1}(\mathbf{u}_{k+1}^*) \le P_{k+1}(\mathbf{u}_k^*). \tag{34}$$

Because, $r_k, k = 1, 2, \ldots$, is an increasing sequence, then

$$P_k(\mathbf{u}_{k+1}^*) \le P_{k+1}(\mathbf{u}_{k+1}^*). \tag{35}$$

It follows from inequalities (33) and (35) that

$$P_k(\mathbf{u}_k^*) \le P_{k+1}(\mathbf{u}_{k+1}^*), \quad \forall k = 1, 2, \ldots, \tag{36}$$

Hence, the proof of Statement 1.

2. Divide both sides of inequalities (33) and (34) by r_k and r_{k+1}, respectively to obtain,

$$\frac{1}{r_k} P_k(\mathbf{u}_k^*) \le \frac{1}{r_k} P_k(\mathbf{u}_{k+1}^*)$$
$$\implies \frac{1}{r_k}\big(f(\mathbf{u}_k^*) + r_k Q(\mathbf{u}_k^*)\big) \le \frac{1}{r_k}\big(f(\mathbf{u}_{k+1}^*) + r_k Q(\mathbf{u}_{k+1}^*)\big)$$
$$\text{(using the definition of } P_k)$$
$$\implies \frac{1}{r_k} f(\mathbf{u}_k^*) + Q(\mathbf{u}_k^*) \le \frac{1}{r_k} f(\mathbf{u}_{k+1}^*) + Q(\mathbf{u}_{k+1}^*) \tag{37}$$

and

$$\frac{1}{r_{k+1}} P_{k+1}(\mathbf{u}_{k+1}^*) \le \frac{1}{r_{k+1}} P_{k+1}(\mathbf{u}_k^*)$$
$$\implies \frac{1}{r_{k+1}}\big(f(\mathbf{u}_{k+1}^*) + r_{k+1} Q(\mathbf{u}_{k+1}^*)\big) \le \frac{1}{r_{k+1}}\big(f(\mathbf{u}_k^*) + r_{k+1} Q(\mathbf{u}_k^*)\big)$$
$$\implies \frac{1}{r_{k+1}} f(\mathbf{u}_{k+1}^*) + Q(\mathbf{u}_{k+1}^*) \le \frac{1}{r_{k+1}} f(\mathbf{u}_k^*) + Q(\mathbf{u}_k^*). \tag{38}$$

Adding inequalities (37) and (38) and rearranging the result, we get

$$\left(\frac{1}{r_k} - \frac{1}{r_{k+1}}\right) f(\mathbf{u}_k^*) \le \left(\frac{1}{r_k} - \frac{1}{r_{k+1}}\right) f(\mathbf{u}_{k+1}^*). \tag{39}$$

Recall from the definition of r_k that for each $k = 1, 2, \ldots$,

$$r_k < r_{k+1} \implies \frac{1}{r_k} > \frac{1}{r_{k+1}} \implies \frac{1}{r_k} - \frac{1}{r_{k+1}} > 0. \tag{40}$$

Using (40), we can easily divide both sides of the inequality (39) by $(\frac{1}{r_k} - \frac{1}{r_{k+1}})$ to obtain

$$f(\mathbf{u}_k^*) \le f(\mathbf{u}_{k+1}^*). \tag{41}$$

Thus the statement 2. holds for any $k = 1, 2, \ldots$,.

3. Adding inequalities (33) and (34) gives

$$P_k(\mathbf{u}_k^*) + P_{k+1}(\mathbf{u}_{k+1}^*) \le P_k(\mathbf{u}_{k+1}^*) + P_{k+1}(\mathbf{u}_k^*)$$
$$\implies P_k(\mathbf{u}_k^*) - P_{k+1}(\mathbf{u}_k^*) - \big(P_k(\mathbf{u}_{k+1}^*) - P_{k+1}(\mathbf{u}_{k+1}^*)\big) \le 0$$
$$\implies r_k Q(\mathbf{u}_k^*) - r_{k+1} Q(\mathbf{u}_k^*) - \big(r_k Q(\mathbf{u}_{k+1}^*) - r_{k+1} Q(\mathbf{u}_{k+1}^*)\big) \le 0$$
$$\implies (r_k - r_{k+1}) Q(\mathbf{u}_{k+1}^*) \ge (r_k - r_{k+1}) Q(\mathbf{u}_k^*). \tag{42}$$

Since, $r_k < r_{k+1} \implies r_k - r_{k+1} < 0$, then dividing both sides of the inequality (42) by $(r_k - r_{k+1})$, a negative value, reverses the inequality sign and hence leads to

$$Q(\mathbf{u}_k^*) \ge Q(\mathbf{u}_{k+1}^*), \tag{43}$$

This completes the proof of Statement 3. Also, this confirms the fact that for any initial infeasible solution $\mathbf{u}_k^*$, the next approximate solution $\mathbf{u}_{k+1}^*$ obtain by the Algorithm 2 is indeed an improved solution (less infeasible), and would eventually lead to feasible solution as $r_k \to +\infty$ as shown in the next theorem. □

Theorem 4.2 (*Convergence Theorem*)**.** *Let $\mathcal{D}$ be the set of feasible solutions to the given constrained optimization problem (5)–(7). Suppose that the penalty function $P_k(\mathbf{u}, r_k)$ is sequentially minimized for a strictly increasing sequence of penalty parameters $r_k, k = 1, 2, \ldots$, i.e., for any fixed k, the optimal solution, $\mathbf{u}_k^*$ of the subproblem (21) becomes the initial starting point for the subproblem at $k = k + 1$, with $r_{k+1} > r_k$. If the sequence of the unconstrained minima, $\{\mathbf{u}_k^*\}_{k=1}^{\infty}$ converges, then its limit is the desired optimum, $\mathbf{u}^* \in \mathcal{D}$ of the original constrained problem (5)–(7) and $Q(\mathbf{u}_k^*) \to 0$ as $k \to +\infty$ respectively.*

Proof. Assume that $\{\mathbf{u}_k^*\}_{k=1}^{\infty}$ has a convergent subsequence whose limit is $\bar{\mathbf{u}}^*$ in $\mathbb{R}^{N_u}$. Hence,

$$\lim_{k\to+\infty} \mathbf{u}_k^* = \bar{\mathbf{u}}^*. \tag{44}$$

We need to show that $\bar{\mathbf{u}}^*$ is feasible to the original constrained optimization problem (5)–(7) and that $\mathbf{u}^* = \bar{\mathbf{u}}^*$. Since for each $k = 1, 2, \ldots$, $\mathbf{u}_k^*$ is the optimal solution of $P_k(u, r_k)$, then the following holds:

$$f(\mathbf{u}_k^*) \le f(\mathbf{u}_k^*) + r_k Q(\mathbf{u}_k^*) = P_k(\mathbf{u}_k^*, r_k) \le P_k(\mathbf{u}^*, r_k) = f(\mathbf{u}^*)$$
$$(\text{since,} \quad Q(\mathbf{u}^*) = 0) \tag{45}$$
$$\implies f(\mathbf{u}_k^*) \le f(\mathbf{u}^*). \tag{46}$$

Also from the inequality (45), we have

$$r_k Q(\mathbf{u}_k^*) \le f(\mathbf{u}^*) - f(\mathbf{u}_k^*) \implies Q(\mathbf{u}_k^*) \le \frac{1}{r_k}\big(f(\mathbf{u}^*) - f(\mathbf{u}_k^*)\big) \tag{47}$$

Taking the limit as $k \to +\infty$ on both sides of the inequality (47), we get

$$\lim_{k\to+\infty} Q(\mathbf{u}_k^*) \le \lim_{k\to+\infty} \frac{1}{r_k}\big(f(\mathbf{u}^*) - f(\mathbf{u}_k^*)\big) = 0 \qquad (\text{since,} \quad \lim_{k\to+\infty} \frac{1}{r_k} = 0)$$
$$\implies \lim_{k\to+\infty} Q(\mathbf{u}_k^*) \le 0 \implies Q(\lim_{k\to+\infty} \mathbf{u}_k^*) \le 0$$
$$\implies Q(\bar{\mathbf{u}}^*) \le 0 \quad (\text{using Eq. (44)}). \tag{48}$$

It follows from (20) and (48) that

$$Q(\bar{\mathbf{u}}^*) = 0. \tag{49}$$

Hence, $\bar{\mathbf{u}}^*$ is feasible (i.e., $\bar{\mathbf{u}}^* \in \mathcal{D}$) to the original problem (5)–(7). By the definition of $\mathbf{u}^*$, then,

$$f(\mathbf{u}^*) \le f(\bar{\mathbf{u}}^*) \tag{50}$$

Since f is continuous and monotone increasing with respect to $\mathbf{u}^*(r_k)$ (according to Statement 3 of Lemma 4.1), taking the limit as $k \to +\infty$ on both sides of (46) gives

$$\lim_{k\to+\infty} f(\mathbf{u}_k^*) = f(\bar{\mathbf{u}}^*) \le f(\mathbf{u}^*) \tag{51}$$

Using inequalities (50) and (51) lead to

$$f(\mathbf{u}^*) = f(\bar{\mathbf{u}}^*). \tag{52}$$

Thus, $\bar{\mathbf{u}}^*$ is the desired optimal solution of the original problem (5)–(7). This concludes the proof of the fact that the sequence $\{\mathbf{u}_k^*\}_{k=1}^{\infty}$ converges to the optimal solution $\mathbf{u}^*$ as $k \to +\infty$. □

5. Applications

In this section, we utilize numerical results to compare the convergence rate and accuracy of the standard EnOpt method with our proposed EPF-EnOpt method. We examine the two optimal solution methods by solving one analytical problem, and one simple water flooding constrained optimization problem.

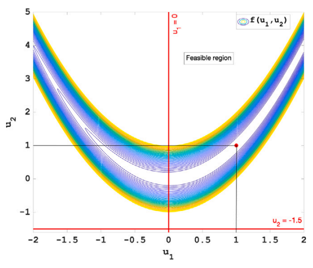


Fig. 1. The contour plot of the objective function in Eq. (53) and its global minimum point in the feasible region.

Example 1

Here, we consider a classical constrained 2D Rosenbrock's optimization problem (also called the Hock & Schittkowski problem 1 (Hock and Schittkowski, 1981; Snyman and Wilke, 2018)) with an additional constraint on the variable u_1. The problem is given as follows.

$$\min_{\mathbf{u}\in\mathbb{R}^2} \; f(\mathbf{u}) = 100(u_2 - u_1^2)^2 + (1 - u_1)^2 \tag{53}$$

$$\text{subject to:} \quad u_1 \geq 0. \tag{54}$$

$$u_2 + 1.5 \geq 0, \tag{55}$$

where $\mathbf{u} = [u_1, u_2]^\top$. By considering a point $\mathbf{u} = [0, 1]^\top$ in the feasible region, we see that the Hessian of the objective function $f(\mathbf{u})$ given by

$$\nabla^2 f(\mathbf{u}) = \begin{bmatrix} 1200u_1^2 - 400u_2 + 2 & -400u_1 \\ -400u_1 & 200 \end{bmatrix} \tag{56}$$

has a negative eigenvalue (and thus, the Hessian of f is not positive definite). This shows that the function is non-convex in the feasible region, and hence in the entire $\mathbb{R}^2$. Because of this, convergence to the global minimizer of the problem (53)–(55) using a numerical minimization procedure can be difficult to achieve. However, an approximation can be found. In theory, the global minimum value of the problem (53)–(55) is at the point $u_1 = 1$ and $u_2 = 1$ as shown in Fig. 1.

First, we apply the standard adaptive EnOpt method of Section 3.1 and the linear transformation of subsection 2.1 to find the solution of problem (53)–(55). In this case, we choose an initial guess $\mathbf{u}_1 = [-2, 0.5]^\top$ (see Fig. 2(b)), the initial step size, β_1^{-1} is set to 0.5. The initial perturbation size for each variable is taken as 0.05 (in other to compute the initial covariance matrix (23)), and the initial step size for covariance adaptation is set to 0.1. Here, we used $N = 100$ perturbation control vectors at each iteration and set $\epsilon_3 = 1.0 \times 10^{-6}$ as the tolerance for convergence. The backtracking parameters, α_1 & α_2 are chosen as 0.5 and 0.001 respectively. The maximum number of trials for the process of backtracking (see Algorithm 1) is set to 20. To use the linear transformation, we set the upper limit, $u_i^{\text{upp}} = 5$ for each variable $u_i, i = 1, 2$. After 600 iterations, the standard EnOpt procedure converges (for which $|f(\mathbf{u}_{l+1}) - f(\mathbf{u}_l)| \leq \epsilon_3 |f(\mathbf{u}_l)|$ is sufficiently satisfied) to the minimum point as shown in Fig. 2(a). Fig. 2(b) further illustrates, on a contour plot, the convergence of the corresponding objective function values (in blue dots) at optimization iterations to its minimum (in red dot) in the feasible region.

However, the large iterations required before convergence to the solution of problem (53)–(55) by using the standard EnOpt method can be significantly reduced by using the proposed EPF-EnOpt method presented in Algorithm 2. To utilize the EPF-EnOpt Algorithm 2 for this example, first, we construct the general term of the penalty function sequence in (15) as follows:

$$\begin{aligned} P_k(\mathbf{u}(r_k)) &= f(\mathbf{u}(r_k)) + r_k Q(\mathbf{u}(r_k)), \qquad k = 1, 2, \dots, \\ &= 100(u_2(r_k) - u_1(r_k)^2)^2 + (1 - u_1(r_k))^2 \\ &\quad + r_k\big((\min\{u_1(r_k), 0\})^2 + (\min\{(u_2(r_k) + 1.5), 0\})^2\big), \end{aligned} \tag{57}$$

where $\mathbf{u}(r_k) = [u_1(r_k), u_2(r_k)]^\top$ denotes a value of $\mathbf{u}$ at a given penalty parameter r_k (Note that, the subproblem associated with (57) according to Eq. (21) shall simply be called the "subproblem P_k"). We set the initial penalty parameter, r_1 and the growth constant, c to 1.5 respectively and choose $\epsilon_1 = 1.0 \times 10^{-6}$ and $\epsilon_2 = 1.0 \times 10^{-6}$. Similarly, we utilize an infeasible initial guess $\mathbf{u}(r_1) = [-2, 0.5]^\top$ (see Fig. 3(b)). For the purpose of results comparison, same initialization (as before) of parameters in the EnOpt procedures are used (in Step 3 of Algorithm 2). In addition, we normalize each variable $u_i(r_k), i = 1, 2$ using the same linear transformation of subsection 2.1 (without truncation) with $u_i^{\text{upp}} = 5, \forall i = 1, 2$. However, we note that normalizing the variables this way is not compulsory as one could use a fixed constant or the maximum of the variables. On the application of Algorithm 2, the resulting successive minimum points (including the initial starting point) of the penalty function terms (or subproblems) in the sequence P_k, $k = 1, 2, 3, \dots, 24$ are presented in Fig. 3(a). Here, we found that, the Algorithm 2 converges to the minimum of problem (53)–(55) after the solution of the subproblem P_{24}. The total number of unconstrained minimization iterations (from the starting point) before convergence is 52 as shown in Fig. 4. Fig. 3(b) depicts the corresponding values (in blue dots) of the objective function (53) at the initial guess and the obtained minimum points of subproblems $\{P_k\}_{k=1}^{24}$ respectively. From the numerical optimization results of this example, we see that the EPF-EnOpt method gives a faster convergence rate and also more accurate results than the standard EnOpt method.

For this example, the large number of iterations required by standard EnOpt method is seen as the effect of the truncation (see Eq. (9)) that often occurs with the perturbed vectors (sampled from a Gaussian distribution with adapted covariance matrix and the approximate solution (at each iteration) as the mean) and the approximate solution respectively. As pointed out in Section 2.1, the truncation affects the quality of the approximate gradient and hence a small step size is required for a decrease of the objective function. This continues at each iteration until convergence and hence the reason for a large number of iterations. Also, with different perturbation size (ranging from 0.0001 to 0.5) for each unknown variable and different updating step sizes for the covariance matrix (ranging from 0.001 to 0.1) we found no improvement in the number of iterations required for the standard EnOpt convergence.

However, in EPF-EnOpt method, infeasible perturbed vectors and iterates are kept. Instead of truncation due to infeasibility (as it is done in the standard EnOpt), the violations of the constraints are handled by the penalty term. By doing this resolves the large number of iterations required by the standard EnOpt to 52.

Example 2

In this case, we consider a practical N_u−dimensional constrained optimization problem of water flooding for a single (i.e we assume no geological uncertainty) 5-Spot oil reservoir (see Fig. 5). This is a synthetic 2D reservoir model with three-phase flow (including oil, water and gas) solved using the OPM-Flow simulator (opm-project.org). The model has a central injection and four production wells spatially distributed in a five-spot pattern as shown in Fig. 5. The reservoir model is uniformly discretized into 50 × 50 grid cells, with $\Delta x = \Delta y = 100$ m. On average, it has approximately 30% porosity with heterogeneous permeability map. The initial reservoir pressure is 200 bar. The initial average oil and water saturations are 0.6546 and 0.3454 respectively (i.e, no free gas). The original oil in place (OOIP) is 4.983×10^6 sm^3.

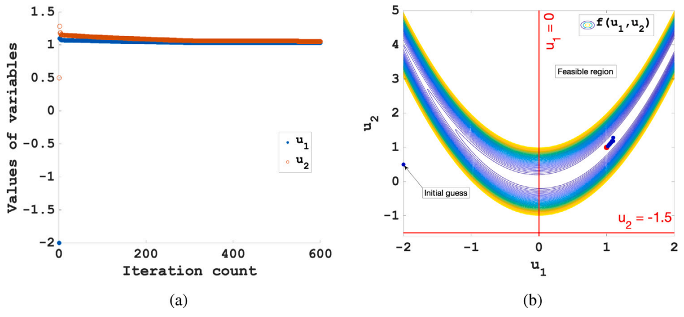


Fig. 2. Minimization of the constrained 2D Rosenbrock problem (53)–(55) using the Standard EnOpt method. (a) Plot of values of control variables versus iteration count. (b) The contour plot of the objective function (53) and its values (in blue dots) at different estimates of the control variables from optimization iterations.

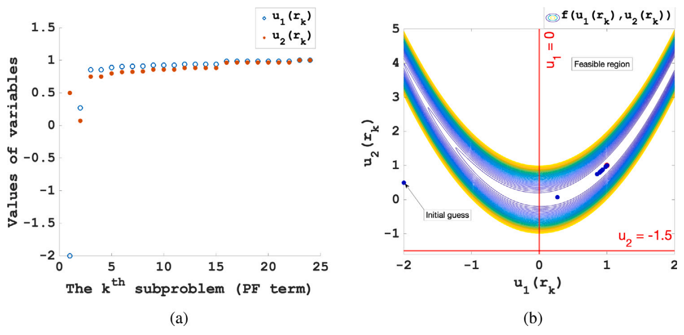


Fig. 3. Minimization of the constrained 2D Rosenbrock problem (53)–(55) using the EPF-EnOpt Algorithm. (a) Plot of the unconstrained minimum points of subproblems in the penalty function sequence. (b) The contour plot of the objective function (53) and its values (in blue dots) at the initial guess and minimum points of subproblems.

Fluid properties is close to that of a light oil reservoir. The viscosity (in cP) for saturated oil at varying bubble point pressure is in the interval $[0.1, 0.56]$ and viscosity of water is 0.01 cP. The densities of oil and water are respectively $732\ \mathrm{kg/m^3}$ and $1000\ \mathrm{kg/m^3}$. The optimization problem is to find the best/ optimum $\mathbf{u} \in \mathbb{R}^{N_u}$ that maximizes the net present value (NPV) defined by the objective function:

$$J(\mathbf{u}) = \sum_{i=1}^{N_t} \frac{r_o Q_{o,i}(\mathbf{u}) + r_g Q_{g,i}(\mathbf{u}) - \left(r_{wi} Q_{w,i}(\mathbf{u}) + r_{wp} Q_{wp,i}(\mathbf{u})\right)}{(1+d_\tau)^{\frac{t_i}{\tau}}}, \tag{58}$$

subject to

$$u_i^{\mathrm{low}} \leq u_i \leq u_i^{\mathrm{upp}}, \quad \forall i = 1, 2, \ldots, N_u, \tag{59}$$

where N_t is the total control time steps; i is the index for time step; d_τ is the discount rate (decimal %) for a given period of time τ (days), and t_i is the cumulative time (days) starting from the beginning of production up to the ith time step; the scalars r_o, r_{wi}, r_{wp}, and r_g denote the price of oil, the cost of handling water injection and production, and the price of gas production (in USD/sm^3) respectively. Let Δt_i be the length of time (days) between t_i and the t_{i-1} time steps. In Eq. (58), $Q_{w,i}$ is the total water injection (sm^3) over the time interval Δt_i; $Q_{o,i}, Q_{wp,i}$, and $Q_{g,i}$ denote the total oil, water, and gas productions (sm^3) over the time interval Δt_i. Also, the quantities $Q_{o,i}, Q_{w,i}, Q_{wp,i}$, and $Q_{g,i}$ are primary variables which depend on the control vector $\mathbf{u}$ at each control time step and their respective numerical values are the results of water flooding reservoir simulation based on a given well configuration data provided by some optimization procedures. We simulated water flooding on the 5-Spot field using the OPM-Flow simulator. In Eq. (59), u_i^{low} and u_i^{upp} are the imposed lower and upper bounds on the control variable u_i, $\forall i = 1, 2, \ldots, N_u$.

In this example, the injection well is controlled by water injection rate at each control time step. The lower and upper bounds for water injection rate are set to 0 sm^3/day and 1000 sm^3/day respectively. Each production well is controlled by reservoir fluid production rate with a lower limit of 0 sm^3/day and upper limit of 250 sm^3/day. Bottom hole pressure (BHP) limits are imposed on the wells, specifically maximum 500 bar for the injector and minimum 150 bar for each producer. The simulation period for the reservoir is 1500 days and the control time step is set to 30 days. Therefore, the optimization unknown control vector $\mathbf{u}$ has a total of $(1+4) \times 50 = 250$ components (control variables). The values of economic parameters utilized for this optimization problem is given by Table 2.

The constrained water flooding optimization (maximization) problem (58)–(59) is treated as a minimization problem by using the transformation in Eq. (4). Next, we seek a solution to the transformed problem, first by using the standard EnOpt approach (see Section 3.1) coupled with the linear transformation (with truncation) of Section 2.1. Here, we select an initial feasible guess, $\mathbf{u}_1$ of control vector (denoted by the "ref control" in Fig. 6). Namely, a constant 500 sm^3/day for the water injection rate at the injection well, and 150 sm^3/day for the reservoir fluid production rate target at each production well. The values of other optimization parameters considered in the EnOpt algorithm

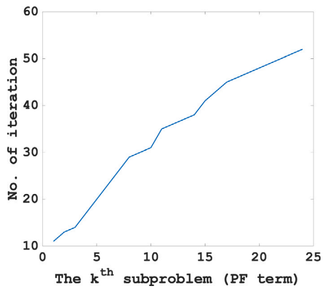


Fig. 4. Plot of cumulative number of unconstrained minimization iterations versus the kth subproblem for the 2D Rosenbrock problem.

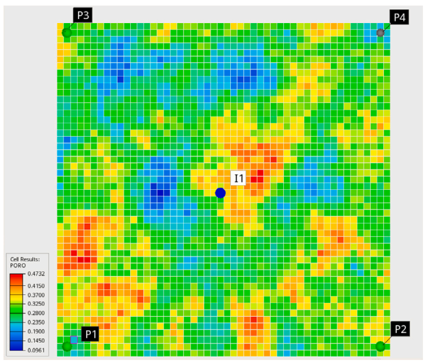


Fig. 5. Porosity distribution of the five-spot field.

are given in Table 1. After 20 iterations, the standard EnOpt algorithm converges (when $|J(\mathbf{u}_{l+1}) - J(\mathbf{u}_l)| \leq \epsilon_3|J(\mathbf{u}_l)|$ is sufficiently satisfied) to a suboptimal solution (represented by the "standard EnOpt" in Fig. 6) of the problem (58)–(59). The maximum NPV reached is 3.855×10^8 (in USD) (see the "standard EnOpt" in Fig. 7(a)).

Furthermore, we used the proposed EPF-EnOpt method (in Algorithm 2) to solve the equivalent minimization problem associated with problem (58)–(59) in other to demonstrate its added advantage in the hydrocarbon industries. First, we rewrite the bound constraint inequalities (59) in the preferred standard form (of inequalities (6)) as follows:

$$g_1(u_i) := u_i - u_i^{low} \geq 0, \quad \text{and} \quad g_2(u_i) := u_i^{upp} - u_i \geq 0, \quad \forall i = 1, 2, \dots, N_u. \tag{60}$$

By using Eqs. (58) and (60), the general PF term associated with problem (58)–(59) is then formulated as follows:

$$\begin{aligned} P_k(\mathbf{u}(r_k)) &= f(\mathbf{u}(r_k)) + r_k Q(\mathbf{u}(r_k)), \qquad k = 1, 2, \dots, \\ &= f(\mathbf{u}(r_k)) \\ &\quad + r_k\left(\sum_{i=1}^{N_u}(\min\{g_1(u_i(r_k)), 0\})^2 + \sum_{i=1}^{N_u}(\min\{g_2(u_i(r_k)), 0\})^2\right), \end{aligned} \tag{61}$$

where $f(\mathbf{u}(r_k)) = -J(\mathbf{u}(r_k))$. Next, we set the initial penalty parameter, r_1 and the growth constant, c to 0.1 and 10 respectively, and $\epsilon_1 = 1.0 \times 10^{-5}$ and $\epsilon_2 = 1.0 \times 10^{-6}$. The initial guess of solution is taken as $\mathbf{u}(r_1) := \mathbf{u}_1$ (see the "Ref control" in Fig. 6). We used the values of parameters in Table 1 required for the unconstrained minimization procedures. We apply the linear transformation (without truncation) of subsection 2.1 to normalize each unknown variable $u_i, i = 1, 2, \dots, N_u$ (and hence the associated constraint function). On the application of Algorithm 2, the resulting limit of the solutions of subproblems P_k, $k = 1, 2, 3, \dots, 26$ (equivalently the solution of problem (58)–(59)) is represented by the "EPF-EnOpt" in Fig. 6. The corresponding value of the objective function (58) at the solution of subproblem, $P_k, k = 1, 2, \dots, 26$ is denoted by the "EPF-EnOpt" in Fig. 7(a). In this case, the Algorithm 2 is seen to converge to a better suboptimal solution of problem (53)–(55) at the 24th subproblem with a total of 51 unconstrained optimization iterations as shown in Fig. 8. The maximum NPV obtained is 4.420×10^8 (in USD). Therefore, the EPF-EnOpt method yields an increase of 14.596% in NPV over the standard

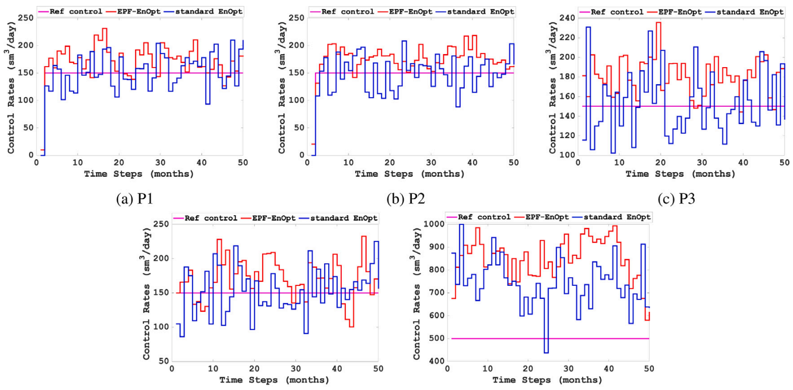
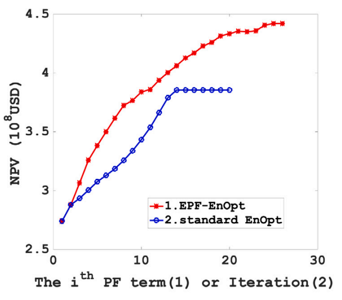
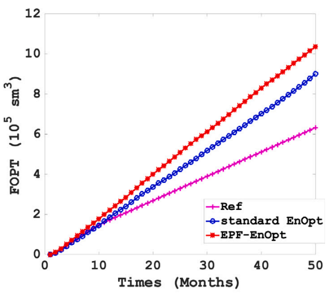


Fig. 6. Comparison of the initial (ref) and optimal solutions from standard EnOpt and EPF-EnOpt methods for the optimization problem (58)–(59). Plots (a) to (d) represent the control (production rate) variation profiles at the four producing wells and (e) denotes the control (injection rate) variation profile at the injection well I1.



(a) NPV variation



(b) Field Oil Production Total

Fig. 7. (a) Comparison of the change in NPV with iteration (for standard EnOpt) or penalty function term (or subproblem). We see that the objective function is flatting out for both methods, which indicate that the stopping criteria is sufficient for this problem. (b) Comparison of the field oil production total (FOPT) from reference (Ref), standard EnOpt, and EPF-EnOpt controls.

Table 1
Standard EnOpt parameters.

Parameter	Value
Initial step size (β_1^{-1})	0.3
Initial stepsize for covariance adaptation (−)	0.1
step size contraction (α_1)	0.5
Constant (α_2)	0.01
Initial control-type variance ($\sigma_m, \forall m = 1, 2, \dots, 5$)	0.001
Number of perturbation control vector (N)	50
Constant correlation factor (ρ)	0.5
Tolerance for EnOpt convergence (ϵ_3)	10^{-6}

Table 2
Economic parameters.

Parameter	Value	Unit
Oil price (r_o)	500	USD/sm^3
Price of gas production (r_g)	0.15	USD/sm^3
Cost of water injection or production (r_{wi}, r_{wp})	30	USD/sm^3
Annual discount rate (d_τ)	0.1	–

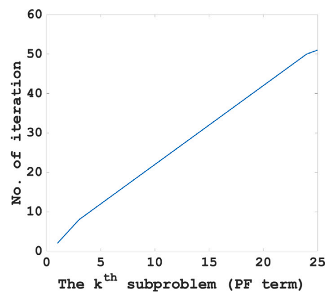


Fig. 8. Plot of cumulative number of unconstrained minimization iteration versus the kth subproblem for the 5-Spot problem.

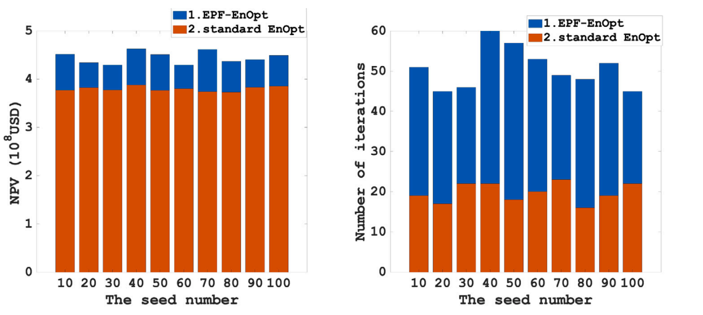
EnOpt approach. Also, Fig. 7(b) compares the field oil production total from operating the field with the solutions (or control strategies) of the EPF-EnOpt and standard EnOpt methods respectively and the reference control. The total field oil production by the standard EnOpt and EPF-EnOpt solutions are 9.006×10^5 sm^3 and 1.036×10^6 sm^3 respectively. It is an increase of approximately 14.983% in oil production. Hence the benefit of the EPF-EnOpt method, as shown in this example is in its better and more accurate result than the traditional EnOpt scheme.

To illustrate that our proposed solution method is not significantly dependent on the state of random number generation (for objective function gradient computation), we compare its performance (in terms of change in NPV and change in cumulative unconstrained iterations) and that of standard EnOpt with different random seed numbers. Fig. 9(a) shows the changes in the maximum NPV obtained from using the EPF-EnOpt and standard EnOpt methods with different seed numbers. On average, the maximum NPV for the EPF-EnOpt and standard EnOpt methods are approximately 4.447×10^8 and 3.857×10^8 (same as initially obtained) respectively. The corresponding total number of iterations carried out with the different seed numbers are shown in Fig. 9(b). The average number of iterations for the standard EnOpt method is 20 and for the EPF-EnOpt method is 51.

Example 3

To further explore the benefit of the proposed EPF-EnOpt method, we consider a more realistic industrial standard 3D oil reservoir model (the Reek field designed by Equinor). The associated N_u−dimensional constrained optimization problem of water flooding associated with the Reek field is given by Eqs. (58)–(59). The reservoir model is synthetic with three-phase flow (including oil, water and gas). It is defined on an irregular grid system of dimension 40×64×14. There are total of 35840 grid cells with distinct sizes. The model is divided into UpperReek, MidReek, and LowerReek zones with six faults and varying porosity and permeability. The model has five production and three injection wells as shown in Fig. 10. Two of the injectors are positioned in the water saturated zones, while the five producers and one injector are spatially distributed throughout the oil containing region based on engineering intuition. Fluid properties are similar to that of light oil. On average, it has approximately 15% porosity with heterogeneous permeability map. The original oil in place (OOIP) is 4.831×10^7 sm^3. The initial average oil, water, and gas saturations are $0.1658, 0.8342$, and 0 respectively. The viscosity (in cP) for saturated oil at varying bubble point pressure is in the interval $[0.09, 1]$ and viscosity of water is 0.01 cP. The densities of oil and water are respectively 732 kg/m^3 and 1000 kg/m^3.

In this example, the injection well is controlled by water injection rate at each control time step. The lower and upper bounds for water injection rate are set to 0 sm^3/day and 10000 sm^3/day respectively. Each production well is controlled by reservoir fluid production rate with a lower limit of 0 sm^3/day and upper limit of 5000 sm^3/day. Bottom hole pressure (BHP) limits are imposed on the wells, specifically maximum 500 bar for the injector and minimum 200 bar for each



(a) NPV variation with random seed (b) Number of iteration variation with random seed

Fig. 9. (a) Comparison of the change in NPV from the standard EnOpt and EPF-EnOpt methods with random seed. (b) Comparison of the total number of iteration from the standard EnOpt and EPF-EnOpt methods with random seed.

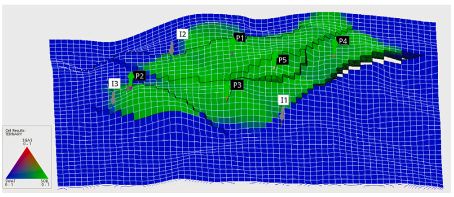


Fig. 10. The initial saturation map for oil, water and gas of the Reek field.

producer. The simulation period for the reservoir is 1110 days and the control time step is set to 30 days. The optimization unknown control vector $\mathbf{u}$ has a total of $(5+3)\times 50 = 296$ components (control variables). Thus, the constrained waterflooding optimization problem is to find the optimal control vector, $\mathbf{u} = \{u_i\}_{i=1}^{296}$ that maximizes the NPV in Eq. (58) subject to the constraints in Eq. (59). Here, the NPV is optimized using the economic parameters listed in Table 2. First, we seek a solution to the optimization problem by using the standard EnOpt approach coupled with the linear transformation (with truncation). In this case, we utilize feasible initial guess, $\mathbf{u}_1$ of control vector (denoted by the "ref control" in Fig. 11) provided by Equinor. The values of other optimization parameters considered in the EnOpt algorithm are given in Table 1.

In 55 iterations, the standard EnOpt algorithm is seen to converge (when $|J(\mathbf{u}_{l+1}) - J(\mathbf{u}_l)| \leq \epsilon_3 |J(\mathbf{u}_l)|$ is sufficiently satisfied) to a suboptimal solution (represented by the "standard EnOpt" in Fig. 11) of the problem (58)–(59). The maximum NPV reached is 4.285×10^9 (in USD) (see the "standard EnOpt" in Fig. 12(a)). Similar to Example 2, we used the proposed EPF-EnOpt method (in Algorithm 2) to solve the subproblems (61) associated with the Reek field waterflooding problem for comparison with the standard EnOpt. We set the initial penalty parameter, r_1 and the growth constant, c to 10 and 1.2 respectively, and $\epsilon_1 = 1.0\times 10^{-5}$ and $\epsilon_2 = 1.0\times 10^{-6}$. The initial guess of solution is taken as $\mathbf{u}(r_1) := \mathbf{u}_1$ (see the "Ref control" in Fig. 11). The same values of parameters in Table 1 are utilized for the unconstrained minimization procedures. Other parameters or transformations not mentioned are the same as before in Example 2. By applying Algorithm 2, the resulting limit of the solutions of subproblems P_k, $k = 1, 2, 3, \dots, 58$ (equivalently the solution of problem (58)–(59)) is denoted by the "EPF-EnOpt" in Fig. 11. The corresponding value of the objective function (58) at the solution of subproblem, $P_k, k = 1, 2, \dots, 58$ is represented by the "EPF-EnOpt" in Fig. 12(a). Here, the Algorithm 2 converges to a better suboptimal solution of problem (58)–(59) at the 59th subproblem with a total of 116 unconstrained optimization iterations as shown in Fig. 13. The maximum NPV obtained is 4.556×10^9 (in USD). Therefore, the EPF-EnOpt method yields an increase of 6.324% in NPV over the standard EnOpt approach.

To further illustrate the efficiency and accuracy of the EPF-EnOpt method, we utilized the classical Lagrange function (LGF) method to handle constraints coupled with the standard EnOpt, to solve the optimization problem and compare results with the EPF-EnOpt. The Lagrange function associated with the constrained water flooding optimization problem of the Reek field (using Eqs. (58)–(59)) is given by :

$$L(\mathbf{u}, \boldsymbol{\lambda}^k, r_k) = f(\mathbf{u}) + r_k\Big(\sum_{i=1}^{N_u}\sum_{j=1}^{2}\Big[(\min\{\frac{\lambda_{j,i}}{2r_k} + g_j(u_i), 0\})^2 - (\frac{\lambda_{j,i}}{2r_k})^2\Big]\Big), \quad (62)$$

where $\boldsymbol{\lambda}^k := [\{\lambda_{j,i}^k\}_{i=1}^{N_u}], \forall i = 1, 2$ is the Lagrangian multiplier for the $2N_u$ constraint functions and r_k is the penalty parameter. For each kth subproblem, we estimate the Lagrangian multiplier using:

$$\boldsymbol{\lambda}^{k+1} := \min\{\boldsymbol{\lambda}^k + 2r_k \mathbf{g}(\mathbf{u}), 0\}, \quad \mathbf{g} = [g_1, g_2], \quad (63)$$

and the penalty parameter r_k is updated using Eq. (17). Using the same optimization parameters, initialization, and convergence tolerance as in the EPF-EnOpt method and initial multiplier components, $\lambda_{j,i} = 0$, we sequentially solve (using the standard EnOpt) for the optimal solution of the Lagrangian in Eq. (62). After 66 subproblems, the Lagrangian converges to a suboptimal value as depicted by "LGF-EnOpt" in Fig. 12(a) with a total of 113 unconstrained iterations as shown by "LGF-EnOpt" in Fig. 13. The limit of solutions of the subproblems is shown by "LGF-EnOpt" in Fig. 11 (equivalently a suboptimal solution to the original constrained optimization problem). The maximum NPV obtained is 4.400×10^9 (in USD) which is less than that of the EPF-EnOpt method by 3.55%. The need for the LGF-EnOpt to estimate the Lagrange multiplier in addition to the penalty parameter shows that the method requires more subproblem iterations.

Fig. 12(b) compares the field oil production total from developing the Reek field with the solutions (or control strategies) of the EPF-EnOpt, LGF-EnOpt, standard EnOpt methods respectively, and the reference control. The total field oil production by the EPF-EnOpt, LGF-EnOpt, and standard EnOpt solutions are 1.091×10^6 sm^3, 1.054×10^6sm^6, and 1.037×10^6 sm^3 respectively. The EPF-EnOpt solution gives an increase of approximately 5.207% and 3.510% in oil production over the standard EnOpt and LGF-EnOpt solutions respectively. Also, we compare the corresponding field water production from the three solutions in Fig. 12(c). The total field water production from the EPF-EnOpt, standard EnOpt, and LGF-EnOpt solutions are 3.338×10^6 sm^3, $3.949 \times$

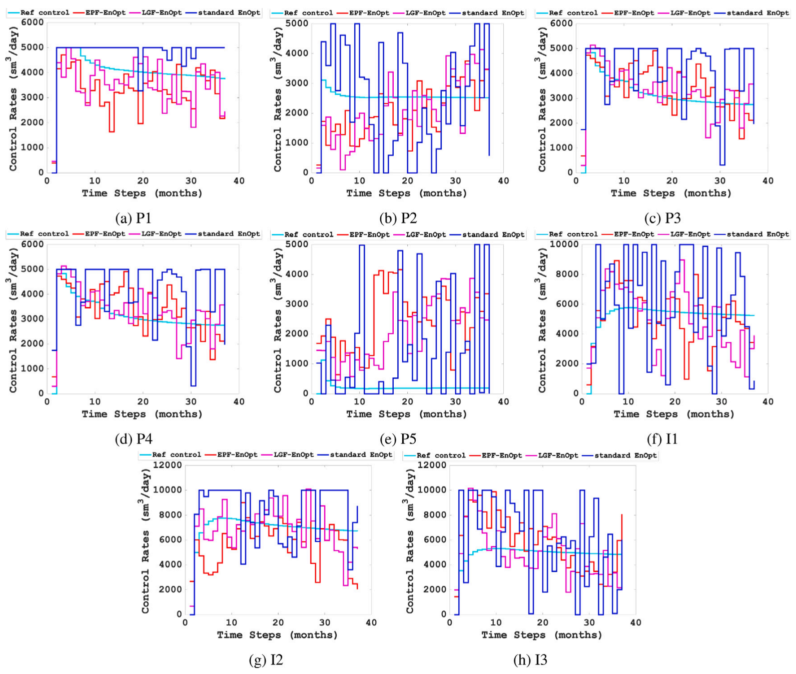


Fig. 11. Comparison of the initial (ref) and optimal solutions from standard EnOpt, EPF-EnOpt, and LGF-EnOpt methods for the constrained optimization problem (58)–(59) associated with the Reek field. Plots (a) to (e) represent the control (production rate) variation profiles at the five producing wells and (f) to (h) denote the control (injection rate) variation profile at the three injection wells.

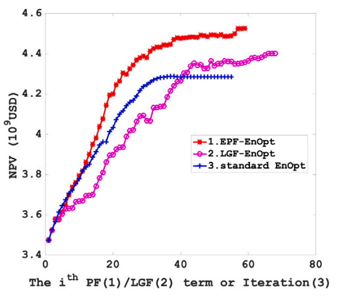
10^6 sm^3, and 3.152×10^6 sm^3 respectively. It is approximately 15.47% and 20.18% in water production by the EPF-EnOpt and LGF-EnOpt solutions less than the standard EnOpt method. Hence, the EPF-EnOpt has the potential to find a better optimal solution that will not only increase oil production but also reduced field water production. This is similar to the solution obtained by using the LGF-method in Zhang et al. (2016).

6. Conclusion

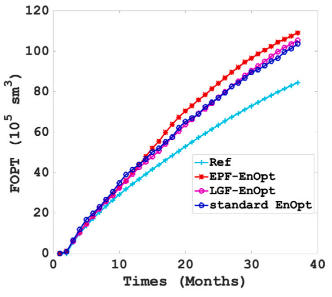
In this study, a new optimal solution strategy, the EPF-EnOpt method, for non-linear constrained optimization problems (such as the one of hydrocarbon field management) is formulated. The strategy leans on the exterior penalty function method and adaptive EnOpt scheme (with backtracking technique). Because of the inappropriate gradient computation arising from using the traditional truncation of control variables to honor of the underlying constraints, we utilized the exterior PF method to transform the constrained optimization problem to a sequence of unconstrained subproblems. We used the adaptive EnOpt method to sequentially solve the subproblems, which eventually leads to the desired solution of the original constrained optimization problem.

Further, the proposed method is formulated in such a way that mixed (involving equality and inequality) constraint problems can be solved efficiently and robustly and also its performance (in terms of convergence rate and accuracy) is compared with the standard EnOpt method. The optimization results of the analytical 2D constrained Rosenbrock problem (see Example 1) showed that the EPF-EnOpt method has a faster convergence rate (and hence more efficient). The solutions of the practical N_u- dimensional constrained water flooding optimization problem associated with the 2D 5Spot field (see Example 2) and 3D Reek field (see Example 3) showed that the method is more accurate than the standard EnOpt approach. In addition to increasing oil production, the EPF-EnOpt methods can provide a solution with less impact (in terms of water production) on the environment compared to the standard EnOpt with traditional constraint handling techniques (as shown in Fig. 12(c)). For the Reek field we also compared the EPF-EnOpt method with the classical Lagrangian constraint handling technique. The results showed higher NPV was obtained with the EPF-EnOpt.

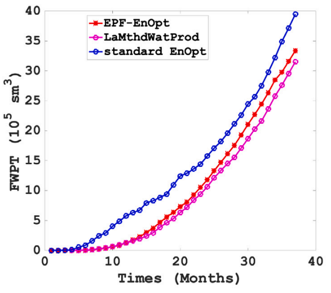
It is noted that the minima of subproblems obtained lie within the feasible region of interest because of our choice of parametric values (irrespective of using infeasible (see Example 1) or feasible (see Example 2) solution initialization). However, this is not always true as different sets of values of parameters such as penalty parameter, growth constant, optimization parameters (such as the number of perturbation and initial step size, and backtracking constants) can change trend of solutions. Also, the size of subproblems and number of optimization iterations before convergence is found to significantly rely on the values assigned to these parameters. Therefore, careful selection of parametric values is the key if strictly feasible solutions are required and for fast convergence rate.



(a) NPV variation



(b) Field Oil Production Total



(c) Field Water Production Total

Fig. 12. (a) Comparison of the change in NPV with iteration (for standard EnOpt) or penalty term or Lagrang e function (or subproblem). (b) Comparison of the field oil production total (FOPT) from reference(Ref), standard EnOpt, and EPF-EnOpt controls.(c) Comparison of the field water production total (FWPT) from standard EnOpt and EPF-EnOpt controls.

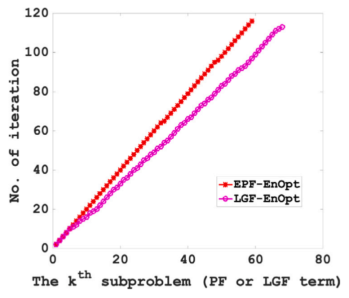


Fig. 13. Plot of cumulative number of unconstrained minimization iteration versus the kth subproblem for the Reek field problem.

For future reference, the present methodology will be used to solve robust constrained optimization problems of oil reservoirs with uncertain geological parameters. In this paper, we considered bound constraint optimization problems. However, solution of optimization problem with mixed or more complex linear or non-linear constraints are possible. Also, there is room to carry out sensitivity analysis with influential parameters on the EPF-EnOpt method.

CRediT authorship contribution statement

Micheal B. Oguntola: Carried out the mathematical formulations and computations of the study, Analyzed the data, Wrote the paper, Contributed to manuscript revisions. **Rolf J. Lorentzen:** Conceived and supervised the study, Contributed to manuscript revisions.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to thank Equinor ASA for developing and sharing the Reek field data. Further, the authors acknowledge the Research Council of Norway and the industry partners, ConocoPhillips Skandinavia AS, Aker BP ASA, Vår Energi AS, Equinor Energy ASA, Neptune Energy Norge AS, Lundin Energy Norway AS, Halliburton AS, Schlumberger Norge AS, Wintershall Dea Norge AS, of The National IOR Centre of Norway for the support. All authors approved the final version of the manuscript.

References

Aanonsen, S.I., Nævdal, G., Oliver, D.S., Reynolds, A.C., Vallès, B., et al., 2009. The ensemble Kalman filter in reservoir engineering–a review. Spe J. 14 (03), 393–412.
Amari, S.-I., 1998. Natural gradient works efficiently in learning. Neural Comput. 10 (2), 251–276. http://dx.doi.org/10.1162/089976698300017746.

Arouri, Y., Sayyafzadeh, M., 2020. An accelerated gradient algorithm for well control optimization. J. Pet. Sci. Eng. 190, 106872.
Bagirov, A., Karmitsa, N., Mäkelä, M.M., 2014. Introduction to Nonsmooth Optimization: Theory, Practice and Software. Springer.
Chen, Y., Oliver, D.S., Zhang, D., 2009. Efficient ensemble-based closed-loop production optimization. SPE J. 14 (04), 634–645.
Chen, Y., Oliver, D.S., et al., 2010. Ensemble-based closed-loop optimization applied to brugge field. SPE Reser. Eval. Eng. 13 (01), 56–71.
Chen, G., Zhang, K., Xue, X., Zhang, L., Yao, J., Sun, H., Fan, L., Yang, Y., 2020. Surrogate-assisted evolutionary algorithm with dimensionality reduction method for water flooding production optimization. J. Pet. Sci. Eng. 185, 106633.
Deb, K., 2012. Optimization for Engineering Design: Algorithms and Examples. PHI Learning Pvt. Ltd.
Do, S.T., Reynolds, A.C., 2013. Theoretical connections between optimization algorithms based on an approximate gradient. Comput. Geosci. 17 (6), 959–973.
Epelle, E.I., Gerogiorgis, D.I., 2020. Adjoint-based well placement optimisation for enhanced oil recovery (EOR) under geological uncertainty: From seismic to production. J. Pet. Sci. Eng. 190, 107091.
Fonseca, R.R.-M., Chen, B., Jansen, J.D., Reynolds, A., 2017. A stochastic simplex approximate gradient (StoSAG) for optimization under uncertainty. Internat. J. Numer. Methods Engrg. 109 (13), 1756–1776.
Fonseca, R.M., Kahrobaei, S.S., Van Gastel, L.J.T., Leeuwenburgh, O., Jansen, J.D., 2015. Quantification of the impact of ensemble size on the quality of an ensemble gradient using principles of hypothesis testing. In: SPE Reservoir Simulation Symposium. Society of Petroleum Engineers.
Fonseca, R., Leeuwenburgh, O., Van den Hof, P., Jansen, J.-D., et al., 2014. Improving the ensemble-optimization method through covariance-matrix adaptation. SPE J. 20 (01), 155–168.
Foroud, T., Baradaran, A., Seifi, A., 2018. A comparative evaluation of global search algorithms in black box optimization of oil production: A case study on Brugge field. J. Pet. Sci. Eng. 167, 131–151.
Han, S.-P., Mangasarian, O.L., 1979. Exact penalty functions in nonlinear programming. Math. Program. 17 (1), 251–269.
Hock, W., Schittkowski, K., 1981. Test Examples for Nonlinear Codes. Springer-Verlag.
Hutahaean, J., Demyanov, V., Christie, M., 2019. Reservoir development optimization under uncertainty for infill well placement in brownfield redevelopment. J. Pet. Sci. Eng. 175, 444–464.
Islam, J., Vasant, P.M., Negash, B.M., Laruccia, M.B., Myint, M., Watada, J., 2020. A holistic review on artificial intelligence techniques for well placement optimization problem. Adv. Eng. Softw. 141, 102767.
Jansen, J., 2011. Adjoint-based optimization of multi-phase flow through porous media - A review. In: 10th ICFD Conference Series on Numerical Methods for Fluid Dynamics (ICFD 2010). Comput. & Fluids 46 (1), 40–51. http://dx.doi.org/10.1016/j.compfluid.2010.09.039.
Jansen, J.-D., Brouwer, R., Douma, S.G., 2009. Closed loop reservoir management. In: SPE Reservoir Simulation Symposium. Society of Petroleum Engineers.
Jesmani, M., Jafarpour, B., Bellout, M.C., Foss, B., 2020. A reduced random sampling strategy for fast robust well placement optimization. J. Pet. Sci. Eng. 184, 106414.
Jung, S., Lee, K., Park, C., Choe, J., 2018. Ensemble-based data assimilation in reservoir characterization: A review. Energies 11 (2), 445.
Li, G., Reynolds, A.C., 2011. Uncertainty quantification of reservoir performance predictions using a stochastic optimization algorithm. Comput. Geosci. 15 (3), 451–462.
Liu, Z., Reynolds, A.C., et al., 2019. An SQP-filter algorithm with an improved stochastic gradient for robust life-cycle optimization problems with nonlinear constraints. In: SPE Reservoir Simulation Conference. Society of Petroleum Engineers.
Lorentzen, R.J., Berg, A., Nævdal, G., Vefring, E.H., 2006. A new approach for dynamic optimization of water flooding problems. In: Intelligent Energy Conference and Exhibition. Society of Petroleum Engineers.
Lu, R., Forouzanfar, F., Reynolds, A.C., et al., 2017. Bi-objective optimization of well placement and controls using stosag. In: SPE Reservoir Simulation Conference. Society of Petroleum Engineers.
Mirzaei-Paiaman, A., Santos, S.M., Schiozer, D.J., 2021. A review on closed-loop field development and management. J. Pet. Sci. Eng. 108457.
Nocedal, J., Wright, S.J., 2006. Numerical Optimization, second ed. Springer.
Oguntola, M., Lorentzen, R., 2020. On the robust value quantification of polymer EOR injection strategies for better decision making. In: ECMOR XVII, 2020. European Association of Geoscientists & Engineers, pp. 1–25.
Ramaswamy, K.R., Fonseca, R.M., Leeuwenburgh, O., Siraj, M.M., Van den Hof, P.M.J., 2020. Improved sampling strategies for ensemble-based optimization. Comput. Geosci. 24 (3), 1057–1069.
Rao, S.S., 2019. Engineering Optimization: Theory and Practice. John Wiley & Sons.
Sarma, P., Aziz, K., Durlofsky, L.J., 2005. Implementation of adjoint solution for optimal control of smart wells. In: SPE Reservoir Simulation Symposium. Society of Petroleum Engineers.
Sarma, P., Chen, W.H., Durlofsky, L.J., Aziz, K., 2006. Production optimization with adjoint models under nonlinear control-state path inequality constraints. In: Intelligent Energy Conference and Exhibition. Society of Petroleum Engineers.
Sarma, P., Chen, W.H., et al., 2008. Efficient well placement optimization with gradient-based algorithms and adjoint models. In: Intelligent Energy Conference and Exhibition. Society of Petroleum Engineers.
Semnani, A., Ostadhassan, M., Xu, Y., Sharifi, M., Liu, B., 2021. Joint optimization of constrained well placement and control parameters using teaching-learning based optimization and an inter-distance algorithm. J. Pet. Sci. Eng. 203, 108652.
Snyman, J.A., Wilke, D.N., 2018. Practical Mathematical Optimization: Basic Optimization Theory and Gradient-Based Algorithms, vol. 133. Springer.
Spall, J.C., 1998. Implementation of the simultaneous perturbation algorithm for stochastic optimization. IEEE Trans. Aerosp. Electron. Syst. 34 (3), 817–823.
Spall, J.C., Hill, S.D., Stark, D.R., 2006. Theoretical framework for comparing several stochastic optimization approaches. In: Probabilistic and Randomized Methods for Design under Uncertainty. Springer, pp. 99–117.
Stordal, A.S., Szklarz, S.P., Leeuwenburgh, O., 2016. A theoretical look at ensemble-based optimization in reservoir management. Math. Geosci. 48 (4), 399–417.
Sun, Y., Liu, Z., Dong, X., 2019. Rate optimization of fractional flow reservoir model based on the continuous adjoint method. J. Pet. Sci. Eng. 183, 106346.
Sun, W., Yuan, Y.-X., 2006. Optimization Theory and Methods: Nonlinear Programming, vol. 1. Springer Science & Business Media.
Xu, L., Zhao, H., Li, Y., Cao, L., Xie, X., Zhang, X., Li, Y., 2018. Production optimization of polymer flooding using improved Monte Carlo gradient approximation algorithm with constraints. J. Circuits Syst. Comput. 27 (11), 1850167.
Yan, X., Reynolds, A.C., 2014. Optimization algorithms based on combining FD approximations and stochastic gradients compared with methods based only on a stochastic gradient. SPE J. 19 (05), 873–890.
Zandvliet, M., Handels, M., van Essen, G., Brouwer, R., Jansen, J.-D., 2008. Adjoint-based well-placement optimization under production constraints. Spe J. 13 (04), 392–399.
Zhang, L., Cui, C., Ma, X., Sun, Z., Liu, F., Zhang, K., 2019. A fractal discrete fracture network model for history matching of naturally fractured reservoirs. Fractals 27 (01), 1940008.
Zhang, Y.T., Lorentzen, R.J., Stordal, A.S., 2018a. Practical use of the ensemble-based conjugate gradient method for production optimization in the brugge benchmark study. In: SPE Norway One Day Seminar. Society of Petroleum Engineers.
Zhang, Y.T., Stordal, A.S., Lorentzen, R.J., Chang, Y., 2018b. A novel ensemble-based conjugate gradient method for reservoir management. In: SPE Norway One Day Seminar. Society of Petroleum Engineers.
Zhang, K., Zhang, X., Ni, W., Zhang, L., Yao, J., Li, L., Yan, X., 2016. Nonlinear constrained production optimization based on augmented Lagrangian function and stochastic gradient. J. Pet. Sci. Eng. 146, 418–431.
Zhao, H., Chen, C., Do, S., Oliveira, D., Li, G., Reynolds, A., et al., 2013. Maximization of a dynamic quadratic interpolation model for production optimization. SPE J. 18 (06), 1–012.
Zhao, X., Zhang, K., Chen, G., Xue, X., Yao, C., Wang, J., Yang, Y., Zhao, H., Yao, J., 2020. Surrogate-assisted differential evolution for production optimization with nonlinear state constraints. J. Pet. Sci. Eng. 194, 107441.
Zhou, K., Hou, J., Zhang, X., Du, Q., Kang, X., Jiang, S., 2013. Optimal control of polymer flooding based on simultaneous perturbation stochastic approximation method guided by finite difference gradient. Comput. Chem. Eng. 55, 40–49.

Paper III

On the Robust Value Quantification of Polymer EOR Injection Strategies for Better Decision Making

Paper IV

Adaptive machine learning based surrogate modeling to accelerate PDE-constrained optimization for enhanced oil recovery

Adaptive machine learning based surrogate modeling to accelerate PDE-constrained optimization in enhanced oil recovery

Tim Keil[3], Hendrik Kleikamp[3], Rolf J. Lorentzen[2], Micheal B. Oguntola[1,2*] and Mario Ohlberger[3]

[1]University of Stavanger, 4036, Stavanger, Norway.
[2]NORCE-Norwegian Research Center AS, 5838, Bergen, Norway.
[3]Institute for Analysis and Numerics and Mathematics Münster, University of Münster, Einsteinstrasse 62, D-48149 Münster.

*Corresponding author(s). E-mail(s): micheal.b.oguntola@uis.no;
Contributing authors: tim.keil@uni-muenster.de; hendrik.kleikamp@uni-muenster.de; rolo@norceresearch.no; mario.ohlberger@uni-muenster.de;

Abstract

In this contribution, we develop an efficient surrogate modeling framework for simulation-based optimization of enhanced oil recovery, where we particularly focus on polymer flooding. The computational approach is based on an adaptive training procedure of a neural network that directly approximates an input-output map of the underlying PDE-constrained optimization problem. The training process thereby focuses on the construction of an accurate surrogate model solely related to the optimization path of an outer iterative optimization loop. True evaluations of the objective function are used to finally obtain certified results. Numerical experiments are given to evaluate the accuracy and efficiency of the approach for a heterogeneous five-spot benchmark problem.

Keywords: PDE-constrained optimization, enhanced oil recovery, machine learning, neural networks, surrogate modeling, ensemble-based optimization

MSC Classification: 49M41 , 68T07 , 90C90

1 Introduction

Water flooding remains the most frequently used secondary oil recovery method. However, the percentage of original oil in place left after the cessation of water flooding in many reservoir fields is estimated to be as high as 50 - 70% [1–3]. The reduced performance of water flooding leading to the sizable leftover of oil has been linked to many factors such as the presence of unfavorable mobility ratios (due to heavy oil), high level of heterogeneity (in porosity and permeability), etc., in the reservoir [4]. For these reasons, enhanced oil recovery (EOR) methods are employed to improve the performance of water flooding in order to increase oil production and minimize environmental stress.

Polymer flooding is a matured chemical EOR method, suitable for heavy oil reservoir development, with over four decades of practical applications [5, 6]. It involves injecting long chains of high-molecular-weight soluble polymers along with water flooding. The polymer EOR mechanism includes reducing mobility ratios of the oil-water system and early water breakthrough in the reservoir by increasing the viscosity of injected water and consequently improving vertical and aerial sweep efficiencies of the injected fluid.

The EOR process of polymer flooding can significantly increase the oil production [6]. However, compared with water flooding, the operational cost and the risk associated with polymer flooding are higher. More so, since injecting more than necessary polymer into the reservoir can lead to insignificant oil increment, it is imperative to optimize the injection strategy of polymer flooding for field application to avoid unnecessarily high operational costs with no profit.

Conventionally, a reservoir simulation model is combined with a numerical optimization technique to determine an optimal control (including water rates, polymer concentrations of injection wells, liquid rates, or bottom hole pressures of production wells) for polymer flooding. The aim is to maximize a given reservoir performance measure (RPM), such as the total oil production or the net present value (NPV) function over the reservoir life. The simulation model is usually a complex numerical reservoir simulator that requires substantial data accounting for geology and geometry of the reservoir or rock and fluid properties. In this study, the model simulates the oil reservoir response (inform of fluid production) to a given polymer flooding control per time. On this account, we estimate the RPM of a given control strategy.

Further, the complexity of a reservoir simulator leads to a high computational effort for simulating a given polymer flooding scenario. It contributes to the inefficiency of gradient-based solution techniques (e.g., the ensemble-based optimization (EnOpt) method) for polymer EOR optimization problems, since the (approximate) gradient of the objective functional with respect to the control variables requires several function evaluations, with each relying on a time-consuming polymer model simulation [7–9]. More so, for large-scale polymer problems discretized into a large number of grid cells, a single model evaluation may take several hours to complete. For this reason, we propose a machine-learning-based approach to approximate the computationally demanding objective functional.

In classical approaches of model order reduction or surrogate modeling, the expensive evaluation of the objective functional due to the PDE constraints is replaced by an a priori trained surrogate model that can be efficiently evaluated with respect to the optimization parameters. In this work, however, we make use of an adaptive surrogate modeling approach, where a surrogate model is constructed during the outer optimization loop through adaptive learning that is targeted towards an accurate input-output map in the vicinity of the chosen parameters during the optimization loop. The overall algorithm thus combines costly full order model (FOM) evaluations, training of machine learning (ML) based surrogate models, as well as evaluations of the successively trained ML models. In model reduction for parameterized systems [10], such adaptive enrichment approaches have been recently proposed and successfully applied in the context of PDE constrained parameter optimization, e.g., in combination with trust-region optimization [11–13]. Recently, in [14, 15] first ideas were presented to combine online enrichment for reduced-order models (ROMs) with machine learning-based surrogate modeling. In this contribution, we use feedforward deep neural networks (DNNs) to obtain surrogate models of the underlying input-output map that directly map the optimization parameters to the output of the objective functional.

Artificial neural networks also gained attention in the context of enhanced oil recovery in recent years, see [16–18], for instance. However, these approaches mainly focus on accelerating the evaluation of the costly objective function without providing a way to solve polymer EOR optimization problems using the proposed surrogate models. In [19], the authors describe an algorithm to obtain a global surrogate model that is applied as a replacement for the objective functional in a genetic algorithm. The global approximation of the objective is computed a priori before applying the optimization routine. In [20], artificial neural networks are employed to facilitate the decision process for a specific EOR method.

Concerning acceleration of PDE-constrained optimization in general, DNNs are, for instance, used in [21] to replace costly simulations within the optimization loops by evaluations of surrogate models. The main idea of the *ISMO* algorithm described in [21] is to run multiple parallel optimization routines starting from different initial guesses and to construct DNN surrogate models using training data collected at the final iterates of these optimization algorithms. The training data is computed by costly evaluations of the exact objective functional (involving the solution of PDEs). In contrast, the optimization routines use the respective surrogate model to speed up the computations. Iteratively, a surrogate model is built to approximate the true objective functional near local optima. The approximation quality also serves as the stopping criterion of the algorithm. Another approach involving physics-informed deep operator networks to accelerate PDE-constrained optimization in a self-supervised manner has recently been suggested in [22].

The idea of not having a global surrogate model, but only approximations of the objective functional that are locally accurate, is also one of the main motivations for our algorithm. In contrast to the procedure in [21] described previously, we iteratively construct DNN surrogate models tailored towards the objective function along a single optimization path. We consider only a single initial guess but check for convergence by taking into account the true objective functional. This stopping criterion certifies that the resulting control is approximately a (local) optimum of the true objective functional and not only of the surrogate. Further, we do not assume that the derivative of the DNN surrogate with respect to its inputs is available but reuse the EnOpt procedure when optimizing with the surrogate model.

The remainder of this article is organized as follows. In Section 2 we introduced the polymer flooding model for EOR and formulate an optimization problem for the economic value of the reservoir response. Section 3 introduces a classical ensemble based optimization algorithm based on a FOM approximation of the polymer flooding model. Feedforward DNNs to approximate the input-output map are introduced in Section 4. In Section 5, we finally present and discuss our new adaptive FOM-ML-based optimization algorithm, which is evaluated numerically for a five-spot benchmark problem in Section 6. Last but not least, a conclusion and outlook is given in Section 7.

2 Optimization of polymer flooding in enhanced oil recovery

The problem of predicting the optimal injection strategy of the polymer EOR method can be formulated as a constrained optimization problem. The setup involves solving a maximization problem in which the objective function, the RPM, is defined on a given set of controllable variables. For the polymer EOR method, a complete set of control variables includes the concentration (and hence volume size of the polymer) and control variables (such as water injection rate, oil production rate, and/or bottom hole pressure for the injecting or producing wells) for water flooding over the producing lifespan of the reservoir.

2.1 Polymer flooding model

As mentioned in the introduction, the optimization process is usually performed on a simulation model of the real reservoir [23]. Here, we consider a polymer flooding simulation model, which is an extension of the black-oil model with a continuity equation for the polymer component [9, 24]. The black-oil model is a special multi-component multi-phase flow model with no diffusion among the fluid components [25]. It assumes that all hydrocarbon species are considered as two components, namely, oil and gas at surface conditions, and can be partially or entirely dissolved in each other to form the oil and gas phases. Further, there is an aqueous phase that consists of only one component called water.

For brevity, we first state the polymer flooding model without mentioning the dependence on the controls and geological parameters explicitly. These dependencies are described in more detail after depicting the model. Hence, in what follows, we assume that fixed sets of controls and geological parameters are given.

In the polymer model, usually, it is assumed that polymer forms an additional component transported in the aqueous phase of the Black-oil model and has no effect on the oil phase. We identify those quantities associated with the water, oil, gas, and polymer components with subscripts W, O, G, and P. In general, the polymer model consists of the following system of partial differential equations:

$$\frac{\partial}{\partial t}(\phi b_{\mathrm{W}} s_{\mathrm{W}}) + \nabla \cdot b_{\mathrm{W}} \mathbf{v}_{\mathrm{W}} = q_{\mathrm{W}}, \tag{1a}$$

$$\frac{\partial}{\partial t}\phi(b_{\mathrm{O}} s_{\mathrm{O}} + r_{\mathrm{OG}} b_{\mathrm{G}} s_{\mathrm{G}}) + \nabla \cdot (b_{\mathrm{O}} \mathbf{v}_{\mathrm{O}} + r_{\mathrm{OG}} b_{\mathrm{G}} \mathbf{v}_{\mathrm{G}}) = q_{\mathrm{O}}, \tag{1b}$$

$$\frac{\partial}{\partial t}\phi(b_{\mathrm{G}} s_{\mathrm{G}} + r_{\mathrm{GO}} b_{\mathrm{O}} s_{\mathrm{O}}) + \nabla \cdot (b_{\mathrm{G}} \mathbf{v}_{\mathrm{G}} + r_{\mathrm{GO}} b_{\mathrm{O}} \mathbf{v}_{\mathrm{O}}) = q_{\mathrm{G}}, \tag{1c}$$

$$\frac{\partial}{\partial t}\Big[\phi(1 - s_{ipv}) s_{\mathrm{W}} + \frac{\rho_r c_a}{b_{\mathrm{W}} c}(1 - \phi)\Big] + \nabla \cdot \mathbf{v}_{\mathrm{P}} = q_{\mathrm{W}}, \tag{1d}$$

where ϕ is the rock porosity, $s_\alpha, b_\alpha, q_\alpha$, and $\mathbf{v}_\alpha$ denote the (unknown) saturation, inverse formation-volume factor (depending on the respective density ρ_α), volumetric source (flow rate per unit volume), and Darcy's flux of phase $\alpha \in \{\mathrm{W}, \mathrm{O}, \mathrm{G}\}$, and r_{OG} and r_{GO} denote the oil-gas and gas-oil ratios. The quantities $\mathbf{v}_{\mathrm{P}}, c_a, s_{ipv}$, and c denote the Darcy's flux, adsorption concentration, inaccessible pore volume, and concentration of the polymer solution, and ρ_r is the density of the reservoir rock.

In addition to the system (1), empirical closure equations for relative permeabilities and capillary pressure in three-phase flow in porous media are applied. Here, the unknown primary variables are phase saturations s_α (or component accumulations) and pressures p_α, and thus, appropriate initial and boundary conditions are defined.

Based on the type of injection and/or production well (e.g., vertical, horizontal, or multi-segment), a suitable well model [26, 27] is coupled with (1) to measure the volumetric flow rates, which depend on the state of the reservoir. A standard well model for vertical wells is given as follows.

The volumetric flow rates q_α for $\alpha \in \{\mathrm{W}, \mathrm{O}, \mathrm{G}\}$ in a multi-phase polymer model are computed using a semi-analytical model according to [26, 28] and are given by

$$q_{\mathrm{W}} = \frac{k_{\mathrm{RW}}(s_{\mathrm{W}})}{\mu_{\mathrm{W,eff}} R_k(c)} WI(p_{\mathrm{bh}} - p_{\mathrm{W}} - \rho_{\mathrm{W}} g(z_{\mathrm{bh}} - z)), \tag{2a}$$

$$q_{\mathrm{O}} = \frac{k_{\mathrm{RO}}(s_{\mathrm{O}})}{\mu_{\mathrm{O,eff}}} WI(p_{\mathrm{bh}} - p_{\mathrm{O}} - \rho_{\mathrm{O}} g(z_{\mathrm{bh}} - z)), \tag{2b}$$

$$q_{\mathrm{G}} = \frac{k_{\mathrm{RG}}(s_{\mathrm{G}})}{\mu_{\mathrm{G,eff}}} WI(p_{\mathrm{bh}} - p_{\mathrm{G}} - \rho_{\mathrm{G}} g(z_{\mathrm{bh}} - z)). \tag{2c}$$

Here, $k_{\mathrm{R}\alpha}(s_\alpha)$, ρ_α, p_α, and $\mu_{\alpha,\mathrm{eff}}$ are the saturation-dependent relative permeability, density, pressure, and effective viscosity of phase $\alpha \in \{\mathrm{W}, \mathrm{O}, \mathrm{G}\}$, WI is the well index, z_{bh} is the well datum level depth, p_{bh} is the bottom hole pressure at the well datum level, z is the depth, $R_k(c)$ models the reduced permeability experienced by the water-polymer mixture, and g is the magnitude of the gravitational acceleration.

Individual wells are usually controlled by surface flow rates or bottom hole pressures. Additional equations which enforce limit values for the component rates and bottom-hole pressures are

$$p_{\mathrm{bh}} - p_{\mathrm{bh}}^{\mathrm{limit}} \leq 0,$$
$$q_\alpha - q_\alpha^{\mathrm{limit}} \leq 0,$$

where $q_\alpha^{\mathrm{limit}}$ is the desired surface-volume rate limit for component α, e.g., field oil rate at the production well, and $p_{\mathrm{bh}}^{\mathrm{limit}}$ is the desired bottom-hole pressure limit. Also, logic constraints to determine what happens if the computed rates or pressures violate the operational constraints, in which case a well may switch from rate control to pressure control, etc., are imposed.

If $q_{\alpha,i}$ is the field volumetric flow rate (in sm^3/day) of component $\alpha \in \{\mathrm{W}, \mathrm{O}, \mathrm{G}\}$ in the production wells over the time interval Δt_i, the field production total (in sm^3) of the component α is given as $Q_{\alpha\mathrm{P},i} = q_{\alpha,i}\Delta t_i$. For polymer production total (in kg), $Q_{\mathrm{PP},i} = c_{\mathrm{L}} q_{\mathrm{W},i}\Delta t_i$, where c_{L} is the leftover field polymer concentration (in kg/sm^3) after adsorption. Injection quantities $Q_{\mathrm{PI},i}$ and $Q_{\mathrm{WI},i}$ are computed similarly, however with volumetric flow rates in the injection wells.

As already mentioned above, the solution of the polymer flooding model stated in (1) depends on a given control vector $\mathbf{u}$, see Section 2.2 for a detailed description of the components of the control vector, and a set of geological properties $\boldsymbol{\theta}$. Consequently, all involved unknowns depend on $\mathbf{u}$ and $\boldsymbol{\theta}$ and the same holds for q_{W}, q_{O}, and q_{G}. From now on, we thus write $Q_{\alpha\mathrm{P},i}(\mathbf{u}, \boldsymbol{\theta})$ for the field production total of component $\alpha \in \{\mathrm{W}, \mathrm{O}, \mathrm{G}\}$, depending on the controls $\mathbf{u}$ and the parameters $\boldsymbol{\theta}$, within the time interval Δt_i, similar as above. We further write $Q_{\mathrm{PP}}(\mathbf{u}, \boldsymbol{\theta})$ for the polymer production total, and $Q_{\mathrm{PI},i}(\mathbf{u}, \boldsymbol{\theta})$ and $Q_{\mathrm{WI},i}(\mathbf{u}, \boldsymbol{\theta})$ for the polymer and water injection.

2.2 Optimization of the economic value of the reservoir response

This study considers the annually discounted net present value (NPV) function as the RPM, similar to the one in [7, 29]. The NPV function is related to the control variables through the polymer simulation model (1). For every polymer control strategy, the NPV function evaluates the economic value of the reservoir response. Also, because the injection and production facilities have limited capacity, the control variables are subject to bound constraints.

Suppose that the geological properties of the oil reservoir of interest, such as porosity, permeability, etc., are known and denoted by $\boldsymbol{\theta}$. Let $\mathcal{D} = \mathbb{R}^{N_u}$ be the domain of control vectors of polymer flooding for the given reservoir, such that

$$\mathbf{u} = \left[u_1^1, u_2^1, \ldots, u_{N_w}^1, \ldots, u_1^{N_t}, u_2^{N_t}, \ldots, u_{N_w}^{N_t}\right]^\mathsf{T},$$

where T means transpose. The subscript of each component of $\mathbf{u}$ denotes the well index, the superscript is the control time step, N_w and N_t denote the number of wells and time steps for each well, respectively, and $N_u = N_w \cdot N_t$ is the total number of control variables. Each component u_j^i in $\mathbf{u}$ represents a control type (e.g., polymer concentration or injection rate, oil or water rate, bottom hole pressure) of well j at the time step i.

The N_u-dimensional optimization problem for polymer flooding is to find the optimal $\mathbf{u} \in \mathcal{D}$ that maximizes the NPV function subject to bound constraints. That is

$$\underset{\mathbf{u}\in\mathcal{D}}{\text{maximize}} \;\; J(\mathbf{u},\boldsymbol{\theta}) := \sum_{i=1}^{N_t} \frac{J_i(\mathbf{u},\boldsymbol{\theta})}{(1+d_\tau)^{\frac{t_i}{\tau}}} \tag{3a}$$

with

$$\begin{aligned} J_i(\mathbf{u},\boldsymbol{\theta}) &:= r_{\mathrm{OP}} Q_{\mathrm{OP},i}(\mathbf{u},\boldsymbol{\theta}) + r_{\mathrm{GP}} Q_{\mathrm{GP},i}(\mathbf{u},\boldsymbol{\theta}) - R_i(\mathbf{u},\boldsymbol{\theta}), \\ R_i(\mathbf{u},\boldsymbol{\theta}) &:= r_{\mathrm{WI}} Q_{\mathrm{WI},i}(\mathbf{u},\boldsymbol{\theta}) + r_{\mathrm{WP}} Q_{\mathrm{WP},i}(\mathbf{u},\boldsymbol{\theta}) \\ &\qquad + r_{\mathrm{PI}} Q_{\mathrm{PI},i}(\mathbf{u},\boldsymbol{\theta}) + r_{\mathrm{PP}} Q_{\mathrm{PP},i}(\mathbf{u},\boldsymbol{\theta}), \end{aligned} \tag{3b}$$

subject to

$$u_j^{\mathrm{low}} \le u_j^i \le u_j^{\mathrm{upp}} \quad \text{for all } j = 1, \ldots, N_w, \; i = 1, \ldots, N_t, \tag{3c}$$

where J_i denotes the cumulative NPV value in the i-th simulation time step. Further, d_τ is the discount rate for a period of τ days, t_i is the cumulative time (in days) starting from the beginning of production up to the i-th time step, and $\Delta t_i := t_i - t_{i-1}$ is the time difference (in days) between the time steps t_i and t_{i-1}. The scalars $r_{\mathrm{OP}}, r_{\mathrm{GP}}, r_{\mathrm{WI}}$ and r_{WP} denote the prices of oil and gas production and the cost of handling water injection and production (in USD/sm^3) respectively, and r_{PI} and r_{PP} are the costs of polymer injection and production (in USD/kg). In addition, $Q_{\mathrm{WI},i}$ and $Q_{\mathrm{PI},i}$ are the total water injection (in sm^3) and total polymer injection or slug size (in kg) over the time interval Δt_i. The quantities $Q_{\mathrm{OP},i}$, $Q_{\mathrm{WP},i}$ and $Q_{\mathrm{GP},i}$ denote the total oil, water and gas productions (in sm^3) over the time interval Δt_i, while $Q_{\mathrm{PP},i}$ represents the total polymer production (in kg) over the time interval Δt_i. The quantities $Q_{\mathrm{OP},i}$, $Q_{\mathrm{WI},i}$, $Q_{\mathrm{WP},i}$, $Q_{\mathrm{GP},i}$, $Q_{\mathrm{PI},i}$, and $Q_{\mathrm{PP},i}$ are computed at each control time step i for given $\mathbf{u}$ and fixed $\boldsymbol{\theta}$ from the polymer flooding model (1) and the well equations (2).

The evaluation of the objective function J in (3a) shall be referred to as the full order model (FOM) function evaluation in the remainder of this study. Therefore, the constrained optimization problem presented in (3) can be interpreted as the FOM optimization problem for polymer flooding, given a suitable discretization of the system (1) (see Section 6.1 for details on the discretization). Also, because $\boldsymbol{\theta}$ is fixed during the optimization process, J is considered a function of $\mathbf{u}$ only, and hence we often write $J(\mathbf{u})$ and $J_i(\mathbf{u})$. The solution method utilized for this optimization problem is presented in the next section.

3 Ensemble based optimization algorithm

In this work, the FOM solution to problem (3) follows from the application of the adaptive ensemble-based optimization (EnOpt) method analogous to the one presented in [7, 30, 31]. We again emphasize that we restrict our attention to a fixed choice of geological parameters $\boldsymbol{\theta}$. Since we apply the EnOpt algorithm later on in our surrogate-based algorithm to a function different from J, we subsequently begin by describing the algorithm in its general form. Afterwards, we discuss the application of the EnOpt algorithm to the objective function J and the resulting computational costs.

3.1 Optimization algorithm for a general objective function

In what follows, we describe the EnOpt algorithm for a general objective function $F\colon \mathbb{R}^{N_u} \to \mathbb{R}$ to iteratively solve the optimization problem

$$\underset{\mathbf{u}\in\mathcal{D}}{\text{maximize}} \;\; F(\mathbf{u}) \tag{4a}$$

$$\text{subject to} \quad u_j^{\text{low}} \leq u_j^i \leq u_j^{\text{upp}} \quad \text{for all } j = 1,\dots,N_w,\; i = 1,\dots,N_t. \tag{4b}$$

The EnOpt method is an iterative method in which one starts with an initial guess $\mathbf{u}_0$ that is usually based on experimental facts in such a way that the underlying constraints in (4b) are satisfied. We sequentially seek for an improved approximate solution $\mathbf{u}$ that maximizes $F(\mathbf{u})$ using a preconditioned (with covariance matrix adaptation) gradient ascent method given by

$$\hat{\mathbf{u}}_{k+1} = \mathbf{u}_k + \beta_k\, \mathbf{d}_k, \tag{5}$$

$$\mathbf{d}_k \approx \frac{\mathbf{C}^k_{\mathbf{u}_k}\mathbf{G}_k}{\|\mathbf{C}^k_{\mathbf{u}_k}\mathbf{G}_k\|_\infty}, \tag{6}$$

where $k = 0, 1, 2, \dots$ is the index of the optimization iteration. The tuning parameter β_k for the step size is computed using an auxiliary line search [32] and is selected such that $0 < \beta_k \leq 1$. Furthermore, $\mathbf{C}^k_{\mathbf{u}_k} \in \mathbb{R}^{N_u \times N_u}$ denotes the user-defined covariance matrix of the control variables at the k-th iteration and $\mathbf{G}_k \in \mathbb{R}^{N_u}$ is the approximate gradient of F with respect to the control variables, preconditioned with $\mathbf{C}^k_{\mathbf{u}_k}$ to obtain the search direction at the k-th iteration.

To ensure that the constraints in (5) are satisfied, the original solution domain of the control variables is projected to the set of admissible controls $\mathcal{D}_{\text{ad}}$, defined as

$$\mathcal{D}_{\text{ad}} \coloneqq \{\mathbf{u} \in \mathcal{D} : u_j^{\text{low}} \leq u_j^i \leq u_j^{\text{upp}} \text{ for all } j = 1, \dots, N_w,\ i = 1, \dots, N_t\}, \quad (7)$$

which corresponds to the constraints in (4b). The updating scheme in (5) is performed in $\mathcal{D}_{\text{ad}}$. We utilize a component-wise projection $P_{\mathcal{D}_{\text{ad}}} : \mathcal{D} \to \mathcal{D}_{\text{ad}}$ on the update $\hat{\mathbf{u}}_{k+1} \in \mathcal{D}$, such that

$$\mathbf{u}_{k+1} = P_{\mathcal{D}_{\text{ad}}}(\hat{\mathbf{u}}_{k+1}) \in \mathcal{D}_{\text{ad}}. \quad (8)$$

In practical applications, it is not common to have controls at different wells to correlate, but the controls may vary smoothly with time at individual wells. Hence, the use of $\mathbf{C}^k_{\mathbf{u}_k}$ in Equation (5) enforces this regularization on the control updates. At $k = 0$, we utilize a temporal covariance function given by

$$\text{Cov}\left(u_j^i, u_j^{i+h}\right) = \sigma_j^2 \rho^h \left(\frac{1}{1-\rho^2}\right), \qquad \text{for all } h \in \{0, \dots, N_t - i\}, \quad (9)$$

from a stationary auto regression of order 1 (i.e., AR(1)) model [33] to compute $\mathbf{C}^0_{\mathbf{u}_0}$ with an assumption that controls of different wells are uncorrelated. The variance for the well j is given by $\sigma_j^2 > 0$, and $\rho \in (-1, 1)$ is the correlation coefficient used to introduce a level of dependence between controls of individual wells at different control time steps (since the AR(1) model is stationary).

The formulation above gives rise to a block diagonal matrix $\mathbf{C}^0_{\mathbf{u}_0}$, which is updated by matrices with rank one at subsequent iterations, using the statistical method presented in [31], to obtain an improved covariance matrix $\mathbf{C}^k_{\mathbf{u}_k}$. For this reason, the solution method in Equation (5) is referred to as the adaptive EnOpt algorithm.

We compute the preconditioned approximate gradient $\mathbf{C}^k_{\mathbf{u}_k}\mathbf{G}_k$ following the approach of the standard EnOpt algorithm. At the k-th iteration, we sample $N \in \mathbb{N}$ control vectors $\mathbf{u}_{k,m} \in \mathcal{D}_{\text{ad}}$, for $m = 1, \dots, N$, from a multivariate Gaussian distribution with mean equal to the k-th control vector $\mathbf{u}_k$ and covariance matrix given by $\mathbf{C}^k_{\mathbf{u}_k}$. Here, the additional subscript m is used to differentiate the perturbed control vectors from the one obtained by Equation (5). The cross-covariance of the control vector $\mathbf{u}_k$ and the objective function $F(\mathbf{u}_k)$ at the k-th iteration is approximated according to [34] as

$$\mathbf{C}^k_{\mathbf{u}_k,F} \coloneqq \frac{1}{N-1} \sum_{m=1}^{N} (\mathbf{u}_{k,m} - \mathbf{u}_k)\big(F(\mathbf{u}_{k,m}) - F(\mathbf{u}_k)\big). \quad (10)$$

Since $\mathbf{u}_{k,m} \sim \mathcal{N}(\mathbf{u}_k, \mathbf{C}^k_{\mathbf{u}_k})$ for $m = 1, \ldots, N$, we assume in Equation (10) that the mean of $\{\mathbf{u}_{k,m}\}_{m=1}^N$ is approximated by $\mathbf{u}_k$. By first-order Taylor series expansion of F about $\mathbf{u}_k$, it can easily be deduced that Equation (10) is an approximation of $\mathbf{C}^k_{\mathbf{u}}\mathbf{G}_k$ at the k-th iteration, that is

$$\mathbf{C}^k_{\mathbf{u}_k}\mathbf{G}_k \approx \mathbf{C}^k_{\mathbf{u}_k,F}, \tag{11}$$

see [30, 35] for a detailed proof. Therefore, we choose the search direction as $\mathbf{d}_k = \mathbf{C}^k_{\mathbf{u}_k,F}/\|\mathbf{C}^k_{\mathbf{u}_k,F}\|_\infty$ in Equation (5). The updating scheme in Equation (5) is performed until the convergence criterion

$$F(\mathbf{u}_k) \leq F(\mathbf{u}_{k-1}) + \varepsilon \tag{12}$$

is satisfied, where $\varepsilon > 0$ is a specified tolerance.

To conclude, for an arbitrary objective function F, the EnOpt procedure described in this section is summarized in Algorithm 1. In this algorithm, the OptStep function replicates a single optimization step in the EnOpt procedure and is detailed in Algorithm 2. We note that returning the set of function values T_{k+1} does not play a role in Algorithm 1 but is crucial for training the surrogate model in Section 5. The line search procedure LineSearch can be found in Algorithm 3.

Algorithm 1 EnOpt algorithm

```
Input: function $F\colon \mathbb{R}^{N_u} \to \mathbb{R}$ for which to solve (4); initial guess $\mathbf{u}_0 \in \mathbb{R}^{N_u}$,
    sample size $N \in \mathbb{N}$, tolerance $\varepsilon > 0$, maximum number of iterations $k^*$,
    initial step size $\beta > 0$, step size contraction $r \in (0,1)$, maximum number
    of step size trials $\nu^* \in \mathbb{N}$
Output: approximate solution $\mathbf{u}^* \in \mathbb{R}^{N_u}$ of (4)
1: function EnOpt$[F](\mathbf{u}_0, N, \varepsilon, k^*, \beta, r, \nu^*)$
2:     $\mathbf{u}_1, T_1 \leftarrow$ OptStep$[F](\mathbf{u}_0, N, 0, \beta, r, \nu^*)$
3:     $k \leftarrow 1$
4:     while $F(\mathbf{u}_k) > F(\mathbf{u}_{k-1}) + \varepsilon$ and $k < k^*$ do
5:         $\mathbf{u}_{k+1}, T_{k+1} \leftarrow$ OptStep$[F](\mathbf{u}_k, N, k, \beta, r, \nu^*)$
6:         $k \leftarrow k + 1$
7:     end while
8:     return $\mathbf{u}^* \leftarrow \mathbf{u}_k$
9: end function
```

Algorithm 2 OptStep algorithm

Input: function $F\colon \mathbb{R}^{N_u} \to \mathbb{R}$; current control vector $\mathbf{u}_k \in \mathbb{R}^{N_u}$, sample size $N \in \mathbb{N}$, number of iteration k, initial step size $\beta > 0$, step size contraction $r \in (0,1)$, maximum number of step size trials $\nu^* \in \mathbb{N}$

Output: update $\mathbf{u}_{k+1} \in \mathbb{R}^{N_u}$ of the controls, set T_{k+1} of N pairs of the form $(\mathbf{u}, F(\mathbf{u}))$

```
1: function OptStep[$F$]($\mathbf{u}_k$, $N$, $k$, $\beta$, $r$, $\nu^*$)
2:     if $k = 0$ then
3:         Compute the initial covariance matrix $\mathbf{C}^0_{\mathbf{u}_0}$ using (9)
4:     else
5:         Compute the covariance matrix $\mathbf{C}^k_{\mathbf{u}_k}$ using the formulation in [31]
6:     end if
7:     Sample $N$ control vectors $\{\mathbf{u}_{k,j}\}_{j=1}^N$ from a distribution $\mathcal{N}(\mathbf{u}_k, \mathbf{C}^k_{\mathbf{u}_k})$
8:     Compute vector $\mathbf{C}^k_{\mathbf{u}_k,F}$ according to (10) and store values $\{F(\mathbf{u}_{k,j})\}_{j=1}^N$
9:     Compute the search direction $\mathbf{d}_k = \mathbf{C}^k_{\mathbf{u}_k,F} / \|\mathbf{C}^k_{\mathbf{u}_k,F}\|_\infty$
10:    $\mathbf{u}_{k+1} \leftarrow$ LineSearch[$F$]($\mathbf{u}_k$, $\mathbf{d}_k$, $\beta$, $r$, $\nu^*$)
11:    $T_{k+1} \leftarrow \{(\mathbf{u}_{k,j}, F(\mathbf{u}_{k,j}))\}_{j=1}^N$
12:    return $\mathbf{u}_{k+1}$, $T_{k+1}$
13: end function
```

Algorithm 3 Line search

Input: function $F\colon \mathbb{R}^{N_u} \to \mathbb{R}$; current controls $\mathbf{u}_k \in \mathbb{R}^{N_u}$, search direction $\mathbf{d}_k \in \mathbb{R}^{N_u}$, initial step size $\beta > 0$, step size contraction $r \in (0,1)$, maximum number of step size trials $\nu^* \in \mathbb{N}$, tolerance $\varepsilon > 0$

Output: update $\mathbf{u}_{k+1} \in \mathbb{R}^{N_u}$ of the controls

```
1: function LineSearch[$F$]($\mathbf{u}_k$, $\mathbf{d}_k$, $\beta$, $r$, $\nu^*$)
2:     $\beta_k \leftarrow \beta$
3:     Compute $\mathbf{u}_{k+1}$ according to (8)
4:     $\nu \leftarrow 0$
5:     while $F(\mathbf{u}_{k+1}) - F(\mathbf{u}_k) \leq \varepsilon$ and $\nu < \nu^*$ do
6:         $\beta_k \leftarrow r\,\beta_k$
7:         Compute $\mathbf{u}_{k+1}$ according to (8)
8:         $\nu \leftarrow \nu + 1$
9:     end while
10:    return $\mathbf{u}_{k+1}$
11: end function
```

3.2 FOM-EnOpt algorithm for enhanced oil recovery

Eventually, we are interested in solving the optimization problem (3) for polymer flooding in enhanced oil recovery. As already discussed in the introduction, our contribution is concerned with the development of a surrogate-based algorithm to reduce the computational costs for solving (3). To this end, if the EnOpt algorithm is used to maximize the function J, defined in Equation (3a), we refer to Algorithm 1 as the *FOM-EnOpt algorithm*. That is, the FOM-EnOpt algorithm is given as EnOpt$[J]$, see Algorithm 4.

Algorithm 4 FOM-EnOpt algorithm

```
Input: initial guess $\mathbf{u}_0 \in \mathbb{R}^{N_u}$, sample size $N \in \mathbb{N}$, tolerance $\varepsilon > 0$, maximum
    number of iterations $k^*$, initial step size $\beta > 0$, step size contraction $r \in$
    $(0, 1)$, maximum number of step size trials $\nu^* \in \mathbb{N}$
Output: approximate solution $\mathbf{u}^* \in \mathbb{R}^{N_u}$ of (3)
1: function FOM-EnOpt($\mathbf{u}_0$, $N$, $\varepsilon$, $k^*$, $\beta$, $r$, $\nu^*$)
2:     return EnOpt$[J]$($\mathbf{u}_0$, $N$, $\varepsilon$, $k^*$ $\beta$, $r$, $\nu^*$)
3: end function
```

As already indicated, we are concerned with the computational effort of the FOM-EnOpt algorithm. Let us recall that evaluating J as in (3a) has the complexity of the high-fidelity reservoir simulator, which, in itself, requires the solution of the discretized polymer flooding model equations (1). In Algorithm 1, the most expensive part is to call OptStep$[J]$, which requires N evaluations of J in Line 8 of Algorithm 2 such that the direction $\mathbf{d}_k$ can be computed in Line 9. Furthermore, the line search in Line 10 evaluates J for every search step. Suppose the simulation time for computing J is particularly large. In that case, the FOM-EnOpt algorithm can be extremely costly, especially if many optimization steps are required since OptStep$[J]$ is called at every iteration step. In this case, all steps in Algorithm 1 and Algorithm 2 that do not require evaluating J are computationally negligible.

Since expensive FOM evaluations are very likely to happen for the presented application, we aim to derive a surrogate-based algorithm that uses an approximation of J whenever possible and thus tries to reduce the number of calls of OptStep$[J]$. Instead, FOM information is reused whenever possible and only computed when necessary. The following section introduces a machine-learning-based way for deriving suitable non-intrusive surrogate models.

4 Neural networks as surrogate model for the input-output map

Deep neural networks (DNNs) are machine learning algorithms suitable for approximating functions without knowing their exact structure. Instead, DNNs can be fitted to approximately reproduce known target values for a set of given inputs. Since DNNs learn from examples of labeled data, they can be seen as *supervised* learning algorithms. In contrast, *unsupervised* machine-learning algorithms try to detect hidden structures within unlabeled data. See [36] for an exhaustive overview of supervised and unsupervised learning algorithms.

A particular class of DNNs are *feedforward neural networks*, in which no cyclic flow of information is allowed. This study considers feedforward neural networks consisting of (fully-connected) linear layers combined with a nonlinear activation function. Our description of these types of DNNs is based on formal definitions that can be found in [37] and [38], for instance.

Feedforward neural networks are used to approximate a given function $f\colon \mathbb{R}^{N_{\text{in}}} \to \mathbb{R}^{N_{\text{out}}}$ for a certain input dimension $N_{\text{in}} \in \mathbb{N}$ and an output dimension $N_{\text{out}} \in \mathbb{N}$. To this end, let $L \in \mathbb{N}$ denote the *number of layers* in the neural network, and $N_{\text{in}} = N_0, N_1, \ldots, N_{L-1}, N_L = N_{\text{out}} \in \mathbb{N}$ the *numbers of neurons* in each layer. Furthermore, the *weights* and *biases* in layer $i \in \{1, \ldots, L\}$ are denoted by $W_i \in \mathbb{R}^{N_i \times N_{i-1}}$ and $b_i \in \mathbb{R}^{N_i}$. We assemble the weights and biases in an L-tuple $\mathbf{W} = \big((W_1, b_1), \ldots, (W_L, b_L)\big)$. Moreover, let $\rho\colon \mathbb{R} \to \mathbb{R}$ be the so-called *activation function* and $\rho_n^*\colon \mathbb{R}^n \to \mathbb{R}^n$ the component-wise application of the activation function ρ for dimension $n \in \mathbb{N}$, that is $\rho_n^*(y) \coloneqq [\rho(y_1), \ldots, \rho(y_n)]^\mathsf{T} \in \mathbb{R}^n$ for $y \in \mathbb{R}^n$. Then we can define the corresponding feedforward neural network in the following way:

Definition 1 (Feedforward neural network) The *feedforward neural network* with weights and biases $\mathbf{W}$ and activation function ρ for approximating $f\colon \mathbb{R}^{N_{\text{in}}} \to \mathbb{R}^{N_{\text{out}}}$, is defined as the function $\Phi_{\mathbf{W}}\colon \mathbb{R}^{N_{\text{in}}} \to \mathbb{R}^{N_{\text{out}}}$. For a given input $x \in \mathbb{R}^{N_{\text{in}}}$, the result $\Phi_{\mathbf{W}}(x) \in \mathbb{R}^{N_{\text{out}}}$ is computed as

$$\Phi_{\mathbf{W}}(x) \coloneqq r_L(x),$$

where $r_L\colon \mathbb{R}^{N_{\text{in}}} \to \mathbb{R}^{N_{\text{out}}}$ is defined in a recursive manner using the functions $r_i\colon \mathbb{R}^{N_{\text{in}}} \to \mathbb{R}^{N_i}$ for $i = 0, \ldots, L-1$, which are given by

$$\begin{aligned} r_L(x) &\coloneqq W_L\, r_{L-1}(x) + b_L, \\ r_i(x) &\coloneqq \rho_{N_i}^*\left(W_i\, r_{i-1}(x) + b_i\right) && \text{for } i = 1, \ldots, L-1, \\ r_0(x) &\coloneqq x. \end{aligned}$$

Fitting neural network weights and biases to a given function f is accomplished by creating a sample set $T_{\text{train}} = \{(x_1, f(x_1)), \ldots, (x_n, f(x_n))\} \subset X \times \mathbb{R}^{N_{\text{out}}}$ (the so-called *training set*), consisting of inputs $x_i \in X$ from an input set $X \subset \mathbb{R}^{N_{\text{in}}}$ and corresponding outputs $f(x_i) \in \mathbb{R}^{N_{\text{out}}}$. The process of finding the weights $\mathbf{W}$ such that $\Phi_{\mathbf{W}}(x_i) \approx f(x_i)$ for $i = 1, \ldots, n$ is called *training* of the neural network. During the training, the weights and biases of the neural network $\Phi_{\mathbf{W}}$ are iteratively adjusted such that a *loss function*, which measures the deviation of the output $\Phi_{\mathbf{W}}(x_i)$ for a given input x_i from the desired result $f(x_i)$, is minimized. A common choice for the loss function is the *mean squared error loss* $\mathcal{L}\left(\Phi_{\mathbf{W}}, T_{\text{train}}\right)$ given as

$$\mathcal{L}\left(\Phi_{\mathbf{W}}, T_{\text{train}}\right) \coloneqq \sum_{(x,y) \in T_{\text{train}}} \|\Phi_{\mathbf{W}}(x) - y\|_2^2. \tag{13}$$

For a fixed *architecture*, i.e. fixed number of layers L and numbers of neurons $N_0, \dots, N_L$ in each layer, we define the set of possible weights and biases Ψ as

$$\Psi := \bigtimes_{i=1}^{L} \left(\mathbb{R}^{N_i \times N_{i-1}} \times \mathbb{R}^{N_i} \right).$$

The set Ψ contains L-tuples such that the matrices and vectors in each tuple have suitable dimensions. The aim of neural network training is to find weights and biases $\mathbf{W}^* \in \Psi$ such that the corresponding function $\Phi_{\mathbf{W}^*}$ minimizes the loss function $\mathcal{L}$, i.e.

$$\mathbf{W}^* = \arg\min_{\mathbf{W} \in \Psi} \mathcal{L}\left(\Phi_{\mathbf{W}}, T_{\text{train}}\right). \tag{14}$$

There are several suitable optimization algorithms to approximate the solution of (14) numerically. All of these methods require access to the gradient of the loss function $\mathcal{L}$ with respect to the weights $\mathbf{W}$ of the DNN, which can be computed efficiently using an algorithm called *backpropagation*, see [39]. Popular examples of optimization algorithms used in neural network training are variants of *(stochastic) gradient descent methods*, see [40] for an overview. For small neural networks with only a few layers and neurons, it is also possible to apply methods that use or approximate higher-order derivatives of the loss function, for instance, the *L-BFGS* optimizer [41], which is a limited-memory variant of the *BFGS* method, see for instance Section 6.1 in [32]. In the context of neural network training, each iteration of the optimizer is called *epoch*. Typically, a maximal number of epochs is prescribed for the optimizer to perform.

To prevent a neural network from *overfitting* the training data, we employ *early stopping* [42]. In this method, the loss function is evaluated on a *validation set* $T_{\text{val}} \subset X \times \mathbb{R}^{N_{\text{out}}}$ after each epoch. The validation set is usually chosen to be disjoint from the training set, i.e. $T_{\text{val}} \cap T_{\text{train}} = \emptyset$. Let $\mathbf{W}_k \in \Psi$ denote the weights in epoch $k \in \mathbb{N}$. In each epoch, the value $\mathcal{L}(\Phi_{\mathbf{W}_k}, T_{\text{val}})$ is computed, and if this value does not decrease anymore over a prescribed number of consecutive epochs, the training is aborted. This method ensures that the resulting neural network can perform well on unseen data (that is assumed to have the same structure as the training data).

The result of the optimization routine typically depends strongly on the initial values $\mathbf{W}_0 \in \Psi$ of the weights. There are several methods for initializing the weights of neural networks, for instance, the so-called Kaiming initialization, see [43] for more details. We perform *multiple restarts* of the training algorithm using different initial values for the weights to minimize the dependence of the resulting neural network on the weight initialization. Finally, we select the neural network $\Phi_{\mathbf{W}^*}$ that produced the smallest loss $\mathcal{L}(\Phi_{\mathbf{W}^*}, T_{\text{train}})+\mathcal{L}(\Phi_{\mathbf{W}^*}, T_{\text{val}})$ over all training restarts, i.e. the smallest combined loss on the training and the validation set.

Finding an appropriate neural network architecture can be difficult in practical applications. Especially the number of layers and the number of neurons significantly influence the approximation capabilities of the resulting neural network. We call a layer *hidden* if it is not an input or an output layer. Neural networks with more than one hidden layer are called *deep neural networks*. See [44] for proofs that DNNs have an increased expressiveness. In addition, there are lots of different activation functions available. Typical examples include the *rectified linear unit (ReLU)* $\rho(x) = \max(x, 0)$, which is nowadays the most popular activation function [45], or the hyperbolic tangent $\rho(x) = \tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$.

5 Adaptive-ML-EnOpt algorithm using deep neural networks

The primary purpose of this work is to propose an adaptive machine-learning-based algorithm for avoiding expensive FOM evaluations as often as possible. To this end, we first discuss the usage of DNNs for the NPV value and subsequently introduce the Adaptive-ML-EnOpt algorithm.

5.1 Surrogate models for the net present value

As discussed in Section 3, we use DNNs to construct a surrogate model for the FOM objective functional J. DNNs are particularly well suited for non-intrusive model reduction if the simulator is considered a black box with no direct access to solutions of the underlying PDEs. In fact, given the formulation of the objective functional (3a), we assume to only have access to the respective components $J_i(\mathbf{u})$.

Following the definition of a DNN in Section 4, two input-output maps can be used to approximate J. We refer to the *scalar-valued output* by considering $J\colon \mathbb{R}^{N_u} \to \mathbb{R}$ as the input-output map. Furthermore, we refer to the *vector-valued output* if we make different use of the structure of J by writing $J(\mathbf{u}) = \delta^\mathsf{T} j(\mathbf{u})$ with

$$\begin{aligned} j\colon \mathbb{R}^{N_u} &\to \mathbb{R}^{N_t}, \\ j(\mathbf{u}) &:= [J_i(\mathbf{u})]_{i=1}^{N_t}, \end{aligned}$$

and the vector $\delta \in \mathbb{R}^{N_t}$, which includes the discount factors, is defined as

$$\delta := \left[\frac{1}{(1+d_\tau)^{\frac{t_i}{\tau}}}\right]_{i=1}^{N_t}.$$

In the scalar-valued case (DNN$_s$-approach), we directly construct a DNN for J with a corresponding function $\Phi_{\mathbf{W}_s}\colon \mathbb{R}^{N_u} \to \mathbb{R}$, i.e. we use a DNN with $N_{\text{in}} = N_u$ and $N_{\text{out}} = 1$. Instead, in the vector-valued case (DNN$_v$-approach), we construct a DNN for approximating j with a corresponding function $\Phi_{\mathbf{W}_v}\colon \mathbb{R}^{N_u} \to \mathbb{R}^{N_t}$ and, by using δ, we indirectly approximate J.

This means that we apply a DNN with input- and output-dimensions given by $N_{\text{in}} = N_u$ and $N_{\text{out}} = N_t$, and multiply the result by δ whenever the respective DNN is used for approximating J. The algorithm described below works for both cases, the scalar-valued and the vector-valued output. Therefore, if access to the individual components of the vector-valued function j is available, it is possible to run the algorithm with both versions. The different neural network output sizes, and therefore, the various structures of the training data, might improve the DNN training results. In our numerical experiment, we observe that the vector-valued DNN yields slightly better results than the scalar-valued DNN (see Section 6). Nevertheless, we consider both the scalar- and vector-valued approaches to discuss the case where the black box reservoir simulator produces only scalar-valued outputs.

By the DNN$_s$- and DNN$_v$-approach, we thus construct a surrogate for the objective function for the optimization problem (3). It remains to explain a suitable and robust EnOpt algorithm that takes advantage of a DNN but shows a similar convergence behavior as the FOM algorithm. A common strategy is to construct a sufficiently accurate surrogate $J_{\text{ML}} \in \{\Phi_{\mathbf{W}_s}, \delta^\mathsf{T}\Phi_{\mathbf{W}_v}\}$ for the entire input space in a large offline time. Following the FOM-EnOpt procedure from Section 3, given J_{ML}, a surrogate-based procedure would then mean to set $F \coloneqq J_{\text{ML}}$ in Algorithm 1. However, no FOM stopping criterion would be used, and since no error control for the surrogate model is given, no certification of the surrogate-based procedure would be available. Importantly, we remark that the input dimension N_u of both DNN approaches is proportional to the number of time steps N_t and the number of physical variables in the model N_w. Thus, dependent on the complexity of the reservoir simulation, N_u may be large. Consequently, it may not be possible to construct a surrogate model with a DNN that is accurate for the entire input space. Even if it were possible to construct such a DNN, we would require prohibitively costly training for computing the training set, validation set, and weights.

5.2 Adaptive algorithm

To circumvent the issue of constructing a globally accurate surrogate, in what follows, we describe the adaptive machine learning EnOpt algorithm (Adaptive-ML-EnOpt). In this algorithm, we incorporate the construction of the DNN into an outer optimization loop trained and certified by FOM quantities. With respect to the FOM-EnOpt procedure, we remark that each FOM optimization step requires N evaluations of J for computing $\mathbf{d}_k$. To obtain an appropriately accurate direction, it is required that N is chosen sufficiently large [46]. For the Adaptive-ML-EnOpt procedure, we only use a single FOM-based optimization step at each outer iteration k. Then, we use the N evaluations of the FOM as data points for training a locally accurate surrogate J^k_{ML}. Instead of proceeding with the FOM functional J, we utilize the DNN to start an inner EnOpt algorithm with $F = J^k_{\text{ML}}$ as objective function in

Section 3.1 and $\mathbf{u}_k$ as initial guess. Denote by $\mathbf{u}_k^{(l)}$ the iterates of the inner optimization loop in the k-th outer iteration, i.e., in particular, we have $\mathbf{u}_k^{(0)} = \mathbf{u}_k$. According to (12), the inner EnOpt iteration terminates if the surrogate-based criterion

$$J_{\mathrm{ML}}^k(\mathbf{u}_k^{(l)}) \leq J_{\mathrm{ML}}^k(\mathbf{u}_k^{(l-1)}) + \varepsilon_i \tag{15}$$

is met for a suitable tolerance $\varepsilon_i > 0$. If the inner iteration terminates after L iterations with a control $\mathbf{u}_k^{(L)}$, the next outer iterate $\mathbf{u}_{k+1}$ is defined as $\mathbf{u}_{k+1} := \mathbf{u}_k^{(L)}$. For a certified FOM-based stopping criterion of the outer optimization loop, given the iterate $\mathbf{u}_k$, we check whether the FOM-EnOpt procedure would, indeed, also stop at the same control point. Thus, we perform a single FOM-based optimization step, which includes the computation of $\mathbf{d}_k$ and the line search, and results in a control $\tilde{\mathbf{u}}_k$. For verifying whether the FOM optimization step successfully finds a sufficiently increasing point at outer iteration k, we consider the FOM termination criterion

$$J(\tilde{\mathbf{u}}_k) \leq J(\mathbf{u}_k) + \varepsilon_o, \tag{16}$$

where $\varepsilon_o > 0$ is a suitable tolerance. If (16) is fulfilled, no improvement of the objective function value using FOM optimization steps can be expected, and therefore we also terminate the Adaptive-ML-EnOpt algorithm. If instead, (16) is not met, we use the computed training data (collected while computing $\mathbf{d}_k$) to retrain the DNN and restart an inner DNN-based EnOpt algorithm. We emphasize that the fully FOM-based stopping criterion constitutes a significant difference to what is proposed in [21], where the termination criterion is based on the approximation quality of the surrogate model at the current iterate. However, we saw in our experiments that such an approximation-based criterion might lead to an undesired early stopping of the algorithm.

One may be concerned about the fact that the surrogate-based inner optimization routine produces a decreasing or stationary point. For this reason, after every outer iteration k of the Adaptive-ML-EnOpt procedure, the inner DNN-optimization is only accepted after a sufficient increase, i.e.

$$J(\mathbf{u}_{k+1}) > J(\mathbf{u}_k) + \varepsilon_o. \tag{17}$$

If an iterate is not accepted, we abort the algorithm. Instead of aborting, one may proceed with an intermediate FOM optimization step. We would further like to emphasize that the fulfillment of (17) also depends on the successful construction of the neural network, meaning that the parameters for the neural network are chosen appropriately. If, instead, (17) fails due to an inaccurate neural network, an automatic variation of the parameters could be enforced to the neural network training, and the corresponding outer iteration should be repeated. However, for the sake of simplicity and because it did not show any relevance in our numerical experiments, we do not specify approaches for the case that $\mathbf{u}_{k+1}$ is not accepted due to (17).

Regarding the choice of the different tolerances ε_i and ε_o for the inner and outer stopping criteria in the Adaptive-ML-EnOpt algorithm, we propose to choose a small value for ε_i similar to the tolerance ε in the FOM-EnOpt procedure. The inner iterations are much cheaper due to the application of a fast surrogate, such that a more significant amount of inner iterations is acceptable. In contrast, we recommend selecting a larger tolerance ε_o to perform fewer outer iterations for obtaining a considerable speed-up. However, if maximum convergence w.r.t. the FOM-EnOpt algorithm is desired, ε_o is to be set equal to ε.

The above-explained Adaptive-ML-EnOpt procedure is summarized in Algorithm 5.

Algorithm 5 Adaptive-ML-EnOpt algorithm

Input: initial guess $\mathbf{u}_0 \in \mathbb{R}^{N_u}$, sample size $N \in \mathbb{N}$, tolerance $\varepsilon_o > 0$ for outer iterations, tolerance $\varepsilon_i > 0$ for inner iterations, maximum number of outer iterations k_o^*, maximum number of inner iterations k_i^*, DNN construction strategy CS $\in \{\text{DNN}_s, \text{DNN}_v\}$, set of DNN-specific variables V_{DNN} as discussed in Section 4 (e.g. network architecture, loss function, training parameters), initial step size $\beta > 0$, step size contraction $r \in (0,1)$, maximum number of step size trials $\nu^* \in \mathbb{N}$

Output: approximate solution $\mathbf{u}^* \in \mathbb{R}^{N_u}$ of (3)

```
1:  function RomEnOpt($\mathbf{u}_0$, $N$, $\varepsilon_o$, $\varepsilon_i$, $k_o^*$, $k_i^*$, CS, $V_{\text{DNN}}$, $\beta$, $r$, $\nu^*$)
2:      $\tilde{\mathbf{u}}_0$, $T_0 \leftarrow$ OptStep[$J$]($\mathbf{u}_0$, $N$, 0, $\beta$, $r$, $\nu^*$)
3:      $k \leftarrow 0$
4:      while $J(\tilde{\mathbf{u}}_k) > J(\mathbf{u}_k) + \varepsilon_o$ and $k < k_o^*$ do
5:          $J_{\text{ML}}^k \leftarrow$ Train($T_k$, CS, $V_{\text{DNN}}$)
6:          $\mathbf{u}_{k+1} \leftarrow$ EnOpt[$J_{\text{ML}}^k$]($\mathbf{u}_k$, $N$, $\varepsilon_i$, $k_i^*$ $\beta$, $r$, $\nu^*$)
7:          if $J(\mathbf{u}_{k+1}) \leq J(\mathbf{u}_k) + \varepsilon_o$ then
8:              return $\mathbf{u}^* \leftarrow \mathbf{u}_k$
9:          end if
10:         $\tilde{\mathbf{u}}_{k+1}$, $T_{k+1} \leftarrow$ OptStep[$J$]($\mathbf{u}_{k+1}$, $N$, $k$, $\beta$, $r$, $\nu^*$)
11:         $k \leftarrow k + 1$
12:     end while
13:     return $\mathbf{u}^* \leftarrow \mathbf{u}_k$
14: end function
```

The Train function performs the neural network training procedure as described in Section 4 and returns, depending on the chosen DNN construction strategy, a function $\Phi_{\mathbf{W}_s}$ or $\Phi_{\mathbf{W}_v}$ that approximates the FOM objective function J. Particularly, the result of Train can be used as the function F in the EnOpt procedure.

The outer acceptance criterion (17) is checked in Line 7. Using the FOM-based stopping criterion in Line 4, we ensure that the Adaptive-ML-EnOpt algorithm has an equivalent stopping procedure as the FOM-EnOpt algorithm, see Line 4 in Algorithm 1. However, the algorithm might terminate at a different (local) optimal point, which we also observe in the numerical experiments.

Compared to the FOM-EnOpt procedure, we emphasize that, in the Adaptive-ML-EnOpt algorithm, mainly the single calls of $\textsc{OptStep}[J]$ in Lines 2 and 10 have FOM complexity, scaling with the number of samples N. Furthermore, the outer stopping criterion in Line 4 and the conditions for acceptance in Line 7 require a single FOM evaluation. The construction of the surrogate makes use of FOM data that is already available from the FOM optimization steps in Lines 2 and 10. In addition, while the training data for Line 5 is available from calling $\textsc{OptStep}[J]$, the training function $\textsc{Train}$ itself is relatively cheap. Furthermore, calling $\textsc{EnOpt}[J_{\mathrm{ML}}^k]$ has low computational effort since evaluating the surrogate J_{ML}^k for a given control (i.e., performing a single forward pass through the neural network) is much faster than evaluating J. The primary motivation for the Adaptive-ML-EnOpt algorithm is the idea that many of the costly FOM optimization steps in the FOM-EnOpt algorithm can be replaced by sequences of cheap calls of $\textsc{EnOpt}[J_{\mathrm{ML}}^k]$ with the surrogate J_{ML}^k. However, since the surrogate might only be reliable in a specific part of the set of feasible control vectors around the current iterate $\mathbf{u}_k$, we retrain the surrogate if the FOM optimization step suggests that a further improvement of the objective function value is possible. Therefore, the overall goal of the Adaptive-ML-EnOpt algorithm is to terminate with a considerably smaller number of (outer) iterations k than the FOM-EnOpt algorithm, and thus, to reduce the computational costs for solving the polymer EOR optimization problem in (3). We refer to the subsequent section for an extensive complexity and run time comparison for a practical example.

The main motivation for the Adaptive-ML-EnOpt algorithm is illustrated in Figure 1. Computing the gradient information using evaluations of the function J is costly, whereas gradient computations using the approximation J_{ML}^k, obtained, for instance, via training a neural network, is cheap. In the example, the Adaptive-ML-EnOpt algorithm performs more optimization steps in total. However, most of these optimization steps are cheap since they only require evaluations of J_{ML}^k. For the Adaptive-ML-EnOpt algorithm, only those steps involving evaluations of J (i.e., outer iterations) require a large computational effort. Each optimization step is costly in the FOM-EnOpt algorithm since the exact objective function J is evaluated multiple times. Altogether, in the example shown in Figure 1, the Adaptive-ML-EnOpt algorithm performs less costly gradient computations than the FOM-EnOpt procedure while arriving approximately at the same optimum. This motivates why the Adaptive-ML-EnOpt algorithm can be preferable with respect to the required computation time.

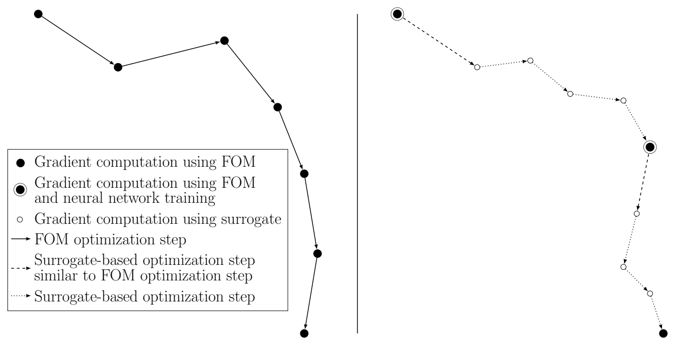

● Gradient computation using FOM
◉ Gradient computation using FOM and neural network training
○ Gradient computation using surrogate
FOM optimization step
Surrogate-based optimization step similar to FOM optimization step
Surrogate-based optimization step

Fig. 1: Example of optimization paths taken by the FOM-EnOpt algorithm (left part of the figure) and the Adaptive-ML-EnOpt algorithm (right part of the figure).

6 Numerical validation for a five-spot benchmark problem

In this section, we present an example with a synthetic oil reservoir in which the polymer flooding optimization problem (3) is solved using the traditional solution method, the FOM-EnOpt algorithm, and our proposed Adaptive-ML-EnOpt method presented in Algorithm 5. The focus is to demonstrate a more efficient and improved method of dealing with the optimization part of a closed-loop reservoir workflow [30] for polymer flooding with the assumption that the geological properties of the reservoir are known. We start by providing information on the algorithm implementation.

6.1 Implementational details

For a numerical approximation of the system (1) of non-linear partial differential equations and the corresponding well equations (2), we make use of the open porous media flow reservoir simulator (OPM) [24, 47]. The system is discretized spatially using a two-point flux approximation (TPFA) with upstream-mobility weighting (UMW) and temporally using a fully-implicit Runge-Kutta method. The resulting discrete-in-time equations are solved using a Newton-Raphson scheme to obtain time-dependent states and the output quantities from the well's equation in terms of fluid production of the reservoir per time step. In this numerical experiment, we perform all polymer flooding simulations in parallel on a 50 core CPU.

For the implementation of the DNN-based surrogates, the Python package pyMOR [48] is used. The implementation of the neural networks and corresponding training algorithms in pyMOR is based on the machine learning library PyTorch [49].

Throughout our numerical experiments described in the subsequent section, we apply the L-BFGS optimizer with strong Wolfe line-search [50, 51] for training the neural networks, i.e., to solve (14). Further, we perform a maximum of 1000 training epochs in each restart.

The number of training restarts influences the accuracy of the trained neural networks and the computation time required for the training. A larger number of restarts typically leads to smaller losses and more training time. To take these two factors into account, we consider different numbers of restarts in our numerical study presented below. The respective results can be found in the subsequent section. In general, we use relatively small numbers of restarts. First of all, we are not interested in obtaining a neural network with very high accuracy. Due to the adaptive retraining of the networks, the surrogates are replaced in each outer iteration anyway. They are only supposed to lead the optimizer to a point with a larger objective function value. On the other hand, as indicated before, a larger number of restarts might result in an unnecessarily long training phase, which must be performed in each outer iteration. The small numbers of 15 and 35 restarts we tried in our studies can thus be seen as a compromise between the accuracy of the surrogate models and computational effort for the training algorithm.

We use 10% of the sample set for validation during the neural network training, and the training routine is stopped early if the loss does not decrease for 10 consecutive epochs. Moreover, the mean squared error loss (MSE loss) is used as the loss function. The neural network training is performed on scaled data. The input values are scaled to $[0,1]^{N_u}$, and the output values are scaled to $[0,1]$ in the DNN$_s$-case and $[0,1]^{N_t}$ in the DNN$_v$-case, respectively. The scaling of the input values can be computed exactly using the lower and upper bounds u_j^{low} and u_j^{upp} for the control variables by

$$u_j^i \mapsto \frac{u_j^i - u_j^{\text{low}}}{u_j^{\text{upp}} - u_j^{\text{low}}} \tag{18}$$

for $j = 1, \dots, N_w$ and $i = 1, \dots, N_t$. For the output values, we take the minimum and maximum value over the training set as lower and upper bound and perform the same scaling as in Equation (18). The tanh function serves as the activation function for each layer. Kaiming initialization is applied for initializing the neural network weights.

The input and output dimensions of the neural networks were already described in Section 5 and are different for the DNN$_s$- and DNN$_v$-case. Regarding the training data for the vector valued case DNN$_v$, we note that we require T_k to store $\{(\mathbf{u}_{k,j}, j(\mathbf{u}_{k,j}))\}_{j=1}^N$ instead of $\{(\mathbf{u}_{k,j}, J(\mathbf{u}_{k,j}))\}_{j=1}^N$, which we did not include in Algorithm 2 for brevity.

6.2 Case study: five-spot field

The numerical experiment considers a two-dimensional reservoir model with a three-phase flow, including oil, water, and gas (cf. Section 2). The computations are performed on a uniform grid that consists of 50×50 grid cells. The model has one injection and four production wells spatially arranged in a five-spot pattern as shown in Figure 2.

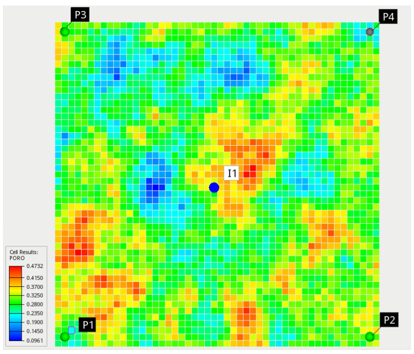


Fig. 2: Porosity distribution of the five-spot field and placement of the injection and production wells.

On average, the reservoir has approximately 30% porosity with a heterogeneous permeability distribution. The initial reservoir pressure is 200 bar. The initial average oil and water saturations are 0.6546 and 0.3454, respectively. The original oil in place is $4.983 \cdot 10^6$ sm^3. Fluid properties are similar to those of a light oil reservoir. The viscosity for saturated oil at varying bubble point pressure lies between 0.1 cP and 0.56 cP, and the viscosity of water is 0.01 cP. The densities of oil and water are taken as 732 kg/m^3 and 1000 kg/m^3, respectively. In this setting, it is easy to see that the displacement is unfavorable since the oil-water mobility ratio λ is such that $10 \leq \lambda \leq 56$. The reservoir rock parameters utilized for the polymer flooding simulation in this problem are given by Table 1.

In this example, the injection well is controlled by two independent control variables, namely the water injection rate and the polymer concentration at each control time step. The lower and upper bounds for the water injection rate are set to 0 sm^3/day and 2000 sm^3/day respectively, while the lower and upper bounds for the polymer concentration are set to 0 kg/sm^3 and 2.5 kg/sm^3. Hence, the polymer injection rate ranges from 0 to 5000 kg/day. Each production well is controlled by a reservoir fluid production rate target with a lower limit of 0 sm^3/day and an upper limit of 500 sm^3/day. Bottom hole pressure limits are imposed on the wells, namely a maximum of 500 bar for the injector and a minimum of 150 bar for each producer. The production

Parameter	Value	Unit
Dead pore space for polymer solution	0.1800	–
Maximum polymer adsorption value	$7.5 \cdot 10^{-4}$	kg/kg
Residual resistance factor of polymer solution	2.5	–
Reservoir rock density	1980	kg/rm^3
Polymer mixing parameters	0.65	–

Table 1: Reservoir model parameters used in the polymer flooding simulations.

period for the reservoir is set to 50 months, and the control time step is taken as 5 months. Therefore, there are $N_u = (2 + 4) \times 10 = 60$ control variables in total to solve for in (3). For the objective function (3a), we used the economic parameters listed in Table 2.

Parameter	Value	Unit
Oil price r_{OP}	500	USD/sm^3
Price of gas production r_{GP}	0.15	USD/sm^3
Cost of polymer injection r_{PI}	2.5	USD/kg
Cost of polymer production r_{PP}	0.5	USD/kg
Cost of water injection or production r_{WI}, r_{WP}	30	USD/sm^3
Annual discount rate d_τ	0.1	–

Table 2: Economic parameters used in the numerical experiments.

Using the two different surrogate models for the objective function (3a) constructed by means of neural networks, namely DNN$_s$ and DNN$_v$ as explained in Section 5, the optimization problem (3) is solved using the Adaptive-ML-EnOpt algorithm. In this case, the Adaptive-ML-EnOpt algorithm for (3) using DNN$_s$ and DNN$_v$ to approximate the objective function J from (3a) is denoted by AML-EnOpt$_s$ and AML-EnOpt$_v$, respectively. The EnOpt parameters for both, the FOM-EnOpt and the two variants of the Adaptive-ML-EnOpt method, are presented in Table 3. We remark that the tolerances ε, ε_i, and ε_o are applied to the scaled quantities, i.e., the output quantities, for which the respective stopping criteria in Algorithms 1 and 5 are checked, have already been scaled as described in Section 6.1.

We compare the Adaptive-ML-EnOpt results with those of the FOM-EnOpt algorithm for two different initial guesses $\mathbf{u}_0^1 \in \mathcal{D}_{\mathrm{ad}}$ and $\mathbf{u}_0^2 \in \mathcal{D}_{\mathrm{ad}}$. The initial solution $\mathbf{u}_0^1$ includes 700 sm^3/day for the water injection rate at the injection well, 150 sm^3/day for the reservoir fluid production rate at each production well, and 0.5 kg/sm^3 for the polymer concentration (equivalently 350 kg/day for polymer injection rate) at the injection well over the simulation period. Similarly, $\mathbf{u}_0^2$ includes 600 sm^3/day for the water injection rate, 100 sm^3/day for the reservoir fluid production rate, and 0.5 kg/sm^3 for the polymer concentration.

Parameter		Value
Initial step size β_0		0.3
Step size contraction r		0.5
Maximum step size trials ν^*		10
Initial control-type variance σ_j		0.001
Constant correlation factor ρ		0.9
Perturbation size N		100
Tolerances	FOM-EnOpt ε	10^{-6}
	Adaptive-ML-EnOpt inner iteration ε_i	10^{-6}
	Adaptive-ML-EnOpt outer iteration ε_o	10^{-2}

Table 3: Parameters used in the FOM-EnOpt and Adaptive-ML-EnOpt algorithms.

Figure 3 compares the values of the objective function during the outer iterations of the FOM-EnOpt, AML-EnOpt$_s$, and AML-EnOpt$_v$ strategies using the initial solutions $\mathbf{u}_0^1$ and $\mathbf{u}_0^2$. Furthermore, the value $J(\mathbf{u}_k)$ at the outer iterate $\mathbf{u}_k$ (denoted by "FOM value") for the respective Adaptive-ML-EnOpt method is depicted.

Since the Adaptive-ML-EnOpt algorithms only use an approximate surrogate model J_{ML}^k, the values of J and J_{ML}^k are not necessarily the same for the control $\mathbf{u}_k$. This behavior is especially apparent in Figure 3(b), where the AML-EnOpt$_v$ algorithm is examined for the initial guess $\mathbf{u}_0^1$. Here, after the first outer iteration, the values $J(\mathbf{u}_1)$ and $J_{\mathrm{ML}}^0(\mathbf{u}_1)$ differ from each other by a significant amount. A possible reason is that the surrogate model J_{ML}^0 does not extrapolate well to the region where the first (inner) Adaptive-ML-EnOpt iteration converged to. This further indicates that the found iterate $\mathbf{u}_1$ is far from the initial solution $\mathbf{u}_0$, where the initial model J_{ML}^0 was trained. However, since the Adaptive-ML-EnOpt algorithm uses evaluations of J in the stopping criterion, the Adaptive-ML-EnOpt does not terminate but continues by training a new surrogate model using training data sampled normally around $\mathbf{u}_1$. Hence, the new surrogate J_{ML}^1 tries to approximate the objective function J well around $\mathbf{u}_1$. In each plot, we see that in the last two outer iterations of the respective Adaptive-ML-EnOpt procedure, the FOM value and the Adaptive-ML-EnOpt value agree to minimal deviations. This suggests that the surrogate model approximates the full objective function well in the region of the (local) optimum found by the Adaptive-ML-EnOpt method.

More so, in Figure 3, it is seen that both, the AML-EnOpt$_s$ and the AML-EnOpt$_v$ algorithm, require considerably less (costly) outer iterations than the FOM-EnOpt method. This leads to an improvement in the run time of the method, which is detailed in Table 4. Besides the faster convergence of the method, we also remark that the Adaptive-ML-EnOpt algorithms find local optima with larger objective function values than the FOM-EnOpt algorithm. However, since the objective function J is multi-modal, this is not guaranteed.

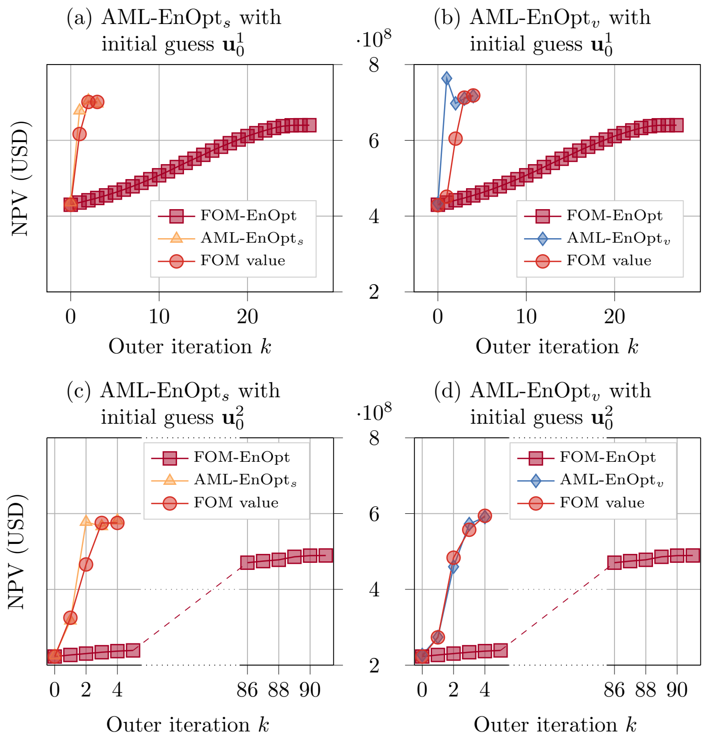


Fig. 3: Comparison of the NPV values obtained during the outer iterations of the FOM-EnOpt, AML-EnOpt$_s$, and AML-EnOpt$_v$ procedures for two different initial guesses $\mathbf{u}_0^1 \in \mathcal{D}_{\mathrm{ad}}$ and $\mathbf{u}_0^2 \in \mathcal{D}_{\mathrm{ad}}$. For each Adaptive-ML-EnOpt procedure, the corresponding FOM value $J(\mathbf{u}_k)$ at the current iterate $\mathbf{u}_k$ of the respective Adaptive-ML-EnOpt method is indicated as well.

We emphasize that each outer iteration of the Adaptive-ML-EnOpt algorithm includes many inner iterations (see also Tables 4 and 5), which leads to the large jumps in the objective function values between consecutive outer iterations, as present in Figure 3.

Further comparisons in terms of function values, numbers of inner and outer iterations, numbers of evaluations of the FOM function J and surrogate approximations J_{ML}^k, total run time, and speedup are presented in Tables 4 and 5.

With the different initial guesses $\mathbf{u}_0^1$ and $\mathbf{u}_0^2$, we found that the number of outer iterations required by the FOM-EnOpt algorithm significantly differs.

However, the Adaptive-ML-EnOpt methods require only 4 and 5 outer iterations. This reduced number of outer iterations leads to a remarkable speedup in the overall computation time T_{total} and is particularly reflected in the reduced number of FOM evaluations, i.e., evaluations of the objective function J, which require costly polymer flooding simulations. Although each outer iteration consists of multiple inner iterations using the surrogate J_{ML}^k, it does not contribute substantially to the overall run time because evaluating the surrogate J_{ML}^k is very cheap.

Method	FOM value	Surrogate value	Outer iter.	Inner iter.	FOM eval.	Surrogate eval.	T_{total} (min)	Speedup
FOM-EnOpt	$6.400 \cdot 10^8$	–	28	–	2839	–	54.86	–
AML-EnOpt$_s$	$7.013 \cdot 10^8$	$6.968 \cdot 10^8$	4	233	407	12315	8.87	6.18
AML-EnOpt$_v$	$7.185 \cdot 10^8$	$7.168 \cdot 10^8$	5	312	509	14101	14.10	3.89

Table 4: Comparisons of the results from the different solution strategies FOM-EnOpt, AML-EnOpt$_s$, and AML-EnOpt$_v$ using the initial guess $\mathbf{u}_0^1$ and $N_1 = N_2 = 35$ neurons in each hidden layer and 15 restarts for the neural network training.

Method	FOM value	Surrogate value	Outer iter.	Inner iter.	FOM eval.	Surrogate eval.	T_{total} (min)	Speedup
FOM-EnOpt	$4.895 \cdot 10^8$	–	92	–	9310	–	135.77	–
AML-EnOpt$_s$	$5.754 \cdot 10^8$	$5.816 \cdot 10^8$	4	111	407	10837	9.85	13.78
AML-EnOpt$_v$	$5.942 \cdot 10^8$	$5.908 \cdot 10^8$	4	117	407	10033	11.05	12.29

Table 5: Comparisons of the results from the different solution strategies FOM-EnOpt, AML-EnOpt$_s$, and AML-EnOpt$_v$ using the initial guess $\mathbf{u}_0^2$ and $N_1 = N_2 = 25$ neurons in each hidden layer and 35 restarts for the neural network training.

For the initial solution $\mathbf{u}_0^1$, the optimizers obtained from the three solution strategies are depicted in Figure 4. Further, the initial guess $\mathbf{u}_0^1$ is shown as a reference.

The control variables obtained by the AML-EnOpt$_s$ and the AML-EnOpt$_v$ algorithm are close to those of the FOM-EnOpt method, except for production well 3 (see Figure 4(c)) and the water injection rate (see Figure 4(e)). For each control variable, the values obtained via the AML-EnOpt$_s$ and AML-EnOpt$_v$ procedures are close to each other. Together with the FOM values of AML-EnOpt$_s$ and AML-EnOpt$_v$ presented in Table 4 and the evolution of the FOM values for the two methods shown in Figure 3(a)-(b), this suggests that the AML-EnOpt$_s$ and the AML-EnOpt$_v$ methods traverse almost the same path in the control space $\mathcal{D}_{\text{ad}}$ and find local optima close to each other.

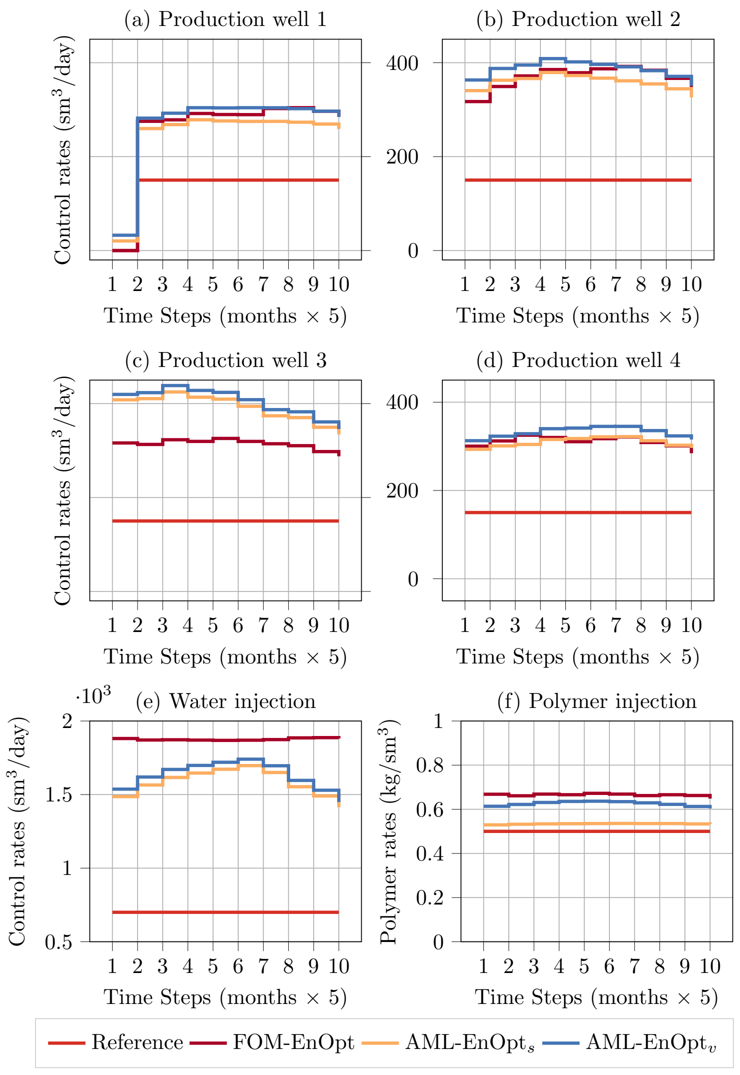


Fig. 4: Comparison of the optimal solutions obtained via the FOM-EnOpt, AML-EnOpt$_s$, and AML-EnOpt$_v$ algorithms using the initial guess $\mathbf{u}_0^1$, which is depicted as the reference solution.

Figure 5(a) depicts a comparison of the total field oil production for the optimal solutions (in Figure 4) of the three solution methods. The total field oil production by FOM-EnOpt, AML-EnOpt$_s$, and AML-EnOpt$_v$ are $1.343 \cdot 10^6$, $1.425 \cdot 10^6$, and $1.554 \cdot 10^6$ (in sm^3), respectively. The solution obtained by AML-EnOpt$_v$ attains the highest oil production in total, followed by the AML-EnOpt$_s$. The total back-produced water and polymer from operating the five-spot field with the different optimal solutions are presented in Figure 5(b) and Figure 5(c), respectively. Here, we found that the AML-EnOpt$_s$ and AML-EnOpt$_v$ solutions are more economical and environmentally friendly than the one provided by using the FOM-EnOpt method.

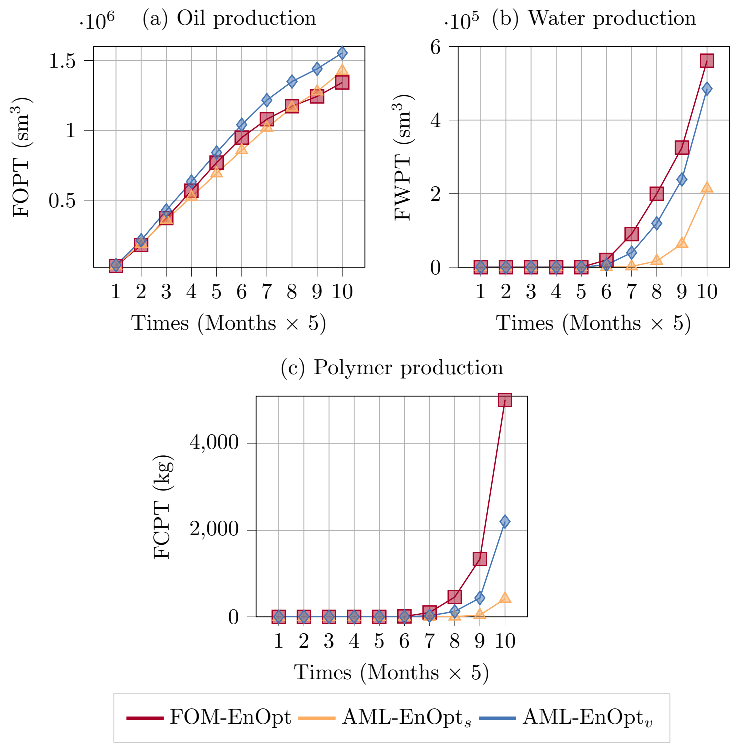


Fig. 5: Comparison of the production data obtained from the different solution strategies FOM-EnOpt, AML-EnOpt$_s$, and AML-EnOpt$_v$ using the initial guess $\mathbf{u}_0^1$.

To further investigate the effects of different neural network architectures on the resulting NPV values, Figure 6 depicts the NPV values obtained by the Adaptive-ML-EnOpt algorithm when using different numbers of neurons in the hidden layers of the surrogate models DNN$_s$ and DNN$_v$.

We observe that the AML-EnOpt$_v$ method results are very similar, which suggests that the DNN$_v$-approach is more robust and leads to similar optimal solutions independent of the neural network structure. In the case of the AML-EnOpt$_s$ algorithm, different numbers of neurons lead to results with a larger variation. In particular, the number of outer iterations performed is different. Hence, the architecture of the underlying network seems to have a significant effect on the performance of the resulting AML-EnOpt$_s$ algorithm.

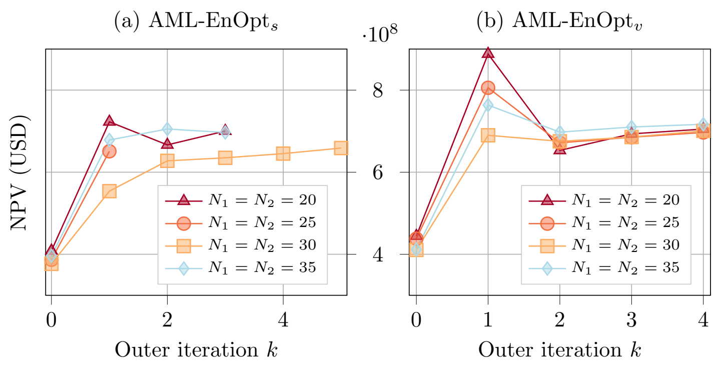


Fig. 6: Comparison of the Adaptive-ML-EnOpt procedures AML-EnOpt$_s$ and AML-EnOpt$_v$ for different numbers of neurons in the hidden layers with fixed initial guess $\mathbf{u}_0^1$.

The maximum, minimum, and average training and validation losses that occurred in the AML-EnOpt$_s$ and AML-EnOpt$_v$ algorithm for the initial guess $\mathbf{u}_0^1$ are presented in Table 6. The table shows the respective MSE losses for different numbers of neurons in the hidden layers.

The results in Table 6 do not suggest a significant influence of the number of neurons on the training and validation results. Further, the scalar- and vector-valued cases, DNN$_s$ and DNN$_v$ respectively, perform similarly in overall training and validation losses. However, we emphasize that, in the DNN$_v$ case, the MSE loss cannot be related directly to the difference in the output function. Instead, one has to take into account that the outputs of DNN$_v$ are summed up to obtain the surrogate J_{ML}^k, while the MSE loss is measured on the vector-valued outputs of the neural network.

Altogether, the numerical experiments with different numbers of neurons suggest that already small DNNs with only 20 neurons in each of the hidden layers yield appropriate results. In this specific application, we do not benefit from increasing the complexity of the neural network. We have seen the same behavior when using more than two hidden layers.

Method	Neurons $N_1 = N_2$	Outer iter.	Training loss			Validation loss		
			Max.	Min.	Avg.	Max.	Min.	Avg.
DNN$_s$	20	4	$1.2 \cdot 10^{-4}$	$1.3 \cdot 10^{-6}$	$5.3 \cdot 10^{-5}$	$5.4 \cdot 10^{-3}$	$6.7 \cdot 10^{-5}$	$2.3 \cdot 10^{-3}$
DNN$_s$	25	2	$6.0 \cdot 10^{-4}$	$1.3 \cdot 10^{-6}$	$3.0 \cdot 10^{-4}$	$2.3 \cdot 10^{-3}$	$2.1 \cdot 10^{-3}$	$2.2 \cdot 10^{-3}$
DNN$_s$	30	7	$7.9 \cdot 10^{-4}$	$8.5 \cdot 10^{-7}$	$1.7 \cdot 10^{-4}$	$6.6 \cdot 10^{-3}$	$1.7 \cdot 10^{-3}$	$3.6 \cdot 10^{-3}$
DNN$_s$	35	4	$1.8 \cdot 10^{-4}$	$6.1 \cdot 10^{-6}$	$8.2 \cdot 10^{-5}$	$5.5 \cdot 10^{-3}$	$3.7 \cdot 10^{-4}$	$2.7 \cdot 10^{-3}$
DNN$_v$	20	5	$1.8 \cdot 10^{-3}$	$2.1 \cdot 10^{-5}$	$5.2 \cdot 10^{-4}$	$6.9 \cdot 10^{-3}$	$1.2 \cdot 10^{-3}$	$4.2 \cdot 10^{-3}$
DNN$_v$	25	6	$9.9 \cdot 10^{-4}$	$1.5 \cdot 10^{-5}$	$4.1 \cdot 10^{-4}$	$6.4 \cdot 10^{-3}$	$9.4 \cdot 10^{-4}$	$3.6 \cdot 10^{-3}$
DNN$_v$	30	5	$9.0 \cdot 10^{-4}$	$9.9 \cdot 10^{-6}$	$4.0 \cdot 10^{-4}$	$1.0 \cdot 10^{-2}$	$5.2 \cdot 10^{-4}$	$4.3 \cdot 10^{-3}$
DNN$_v$	35	5	$6.0 \cdot 10^{-4}$	$1.1 \cdot 10^{-6}$	$2.2 \cdot 10^{-4}$	$8.8 \cdot 10^{-3}$	$4.6 \cdot 10^{-4}$	$4.3 \cdot 10^{-3}$

Table 6: Maximum, minimum, and average MSE loss in the AML-EnOpt$_s$ and AML-EnOpt$_v$ algorithm with different numbers of neurons in the hidden layers of the neural networks DNN$_s$ and DNN$_v$ for fixed initial guess $\mathbf{u}_0^1$. The number of hidden layers is fixed to two.

7 Conclusion and future work

In this contribution, we presented a new algorithm to speed up PDE-constrained optimization problems occurring in the context of enhanced oil recovery. The algorithm is based on adaptively constructed surrogate models that make use of deep neural networks for approximating the objective functional. In each outer iteration of the algorithm, a new surrogate model is trained with data consisting of full-order function evaluations around the current control point. Afterwards, an ensemble-based optimization algorithm is applied to the surrogate to obtain a candidate for the next iteration. We perform full order model evaluations to validate whether the resulting controls correspond to a local optimum of the true objective functional. These function evaluations also serve as training data for constructing the next surrogate.

Our numerical results confirm that the described algorithm can accelerate the solution of the enhanced oil recovery optimization problem. At the same time, in our numerical experiments, the procedure produces controls with even larger objective function values than those obtained using only costly full-order model evaluations. However, we should emphasize that such an improvement in the objective function value is not guaranteed and, in our case, results from the multi-modal structure of the objective functional.

The investigated five-spot benchmark problem served as a proof of concept for our Adaptive-ML-EnOpt algorithm, where FOM evaluations were relatively quickly accessible, and the input dimension was of moderate size. Future research is thus devoted to more involved numerical experiments with more significant complexity.

As indicated in the optimization problem description, we focused on a scenario with fixed geological properties. However, in practical applications, these geological parameters are usually unknown and typically treated by ensemble-based methods, where the ensemble is to be understood not only with respect to perturbations of the controls for approximating the gradient but also with respect to different samples of geological properties. One of the central future research perspectives is incorporating such geological uncertainty in our algorithm. The main challenge is the high dimension of the space of possible geological parameters. Naively using these parameters as additional inputs for the neural network is thus not feasible. Future research might consider reducing the dimension of the space of geological parameters by incorporating additional information on the distribution of such parameters and passing the reduced variables to the neural networks.

Furthermore, replacing neural networks as surrogate models for the objective function, for instance, by polynomial approximations obtained via linear regression or by different machine learning approaches, such as kernel methods [52], could be investigated further. The Adaptive-ML-EnOpt algorithm is formulated in such a way that replacing the surrogate model and its training is readily possible. Any approximation of the objective function built from evaluations of the true objective function is feasible and can directly be used in the algorithm. In addition, the inner iterations are not restricted to the EnOpt procedure but can also be performed using different optimization routines. However, we should emphasize that in the current formulation, no information on the exact gradient, neither of the true objective functional nor the surrogate model, is required. This might change when employing different optimization routines. Moreover, the presented approach is not restricted to the NPV objective functional in enhanced oil recovery but can be generalized to any scalar-valued quantity of interest. The algorithm might be of particular relevance in cases where no direct access to the underlying PDE solutions is possible, and no error estimation for the surrogate model is available.

Acknowledgement

- Tim Keil, Hendrik Kleikamp, Micheal Oguntola and Mario Ohlberger acknowledge funding by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy EXC 2044 –390685587, Mathematics Münster: Dynamics–Geometry–Structure.
- Tim Keil and Mario Ohlberger acknowledge funding by the Deutsche Forschungsgemeinschaft under contract OH 98/11-1.
- Micheal Oguntola and Rolf Lorentzen acknowledge funding from the Research Council of Norway and the industry partners, ConocoPhillips Skandinavia AS, Aker BP ASA, Vår Energi AS, Equinor Energy AS, Neptune Energy Norge AS, Lundin Energy Norway AS, Halliburton AS, Schlumberger Norge AS, and Wintershall Dea Norge AS, of The National IOR Centre of Norway.

Declarations

Conflict of interest The authors declare no competing interests.

References

[1] Van, S.L., Chon, B.H.: Well-pattern investigation and selection by surfactant-polymer flooding performance in heterogeneous reservoir consisting of interbedded low-permeability layer. Korean Journal of Chemical Engineering **33**(12), 3456–3464 (2016). https://doi.org/10.1007/s11814-016-0190-7

[2] Pancholi, S., Negi, G.S., Agarwal, J.R., Bera, A., Shah, M.: Experimental and simulation studies for optimization of water–alternating-gas (co2) flooding for enhanced oil recovery. Petroleum Research **5**(3), 227–234 (2020). https://doi.org/10.1016/j.ptlrs.2020.04.004

[3] Zhang, Y., Lu, R., Forouzanfar, F., Reynolds, A.C.: Well placement and control optimization for wag/sag processes using ensemble-based method. Computers & Chemical Engineering **101**, 193–209 (2017). https://doi.org/10.1016/j.compchemeng.2017.02.020

[4] Gudiña, E.J., Fernandes, E.C., Rodrigues, A.I., Teixeira, J.A., Rodrigues, L.R.: Biosurfactant production by bacillus subtilis using corn steep liquor as culture medium. Frontiers in microbiology **6**, 59 (2015). https://doi.org/10.3389/fmicb.2015.00059

[5] Abidin, A., Puspasari, T., Nugroho, W.: Polymers for enhanced oil recovery technology. Procedia Chemistry **4**, 11–16 (2012). https://doi.org/10.1016/j.proche.2012.06.002

[6] Wang, D., Seright, R.S., Shao, Z., Wang, J., *et al.*: Key aspects of project design for polymer flooding at the daqing oilfield. SPE Reservoir Evaluation & Engineering **11**(06), 1–117 (2008). https://doi.org/10.2118/109682-PA

[7] Oguntola, M.B., Lorentzen, R.J.: On the robust value quantification of polymer eor injection strategies for better decision making. In: ECMOR XVII, vol. 2020, pp. 1–25 (2020). https://doi.org/10.3997/2214-4609.202035057. European Association of Geoscientists & Engineers

[8] Xu, L., Zhao, H., Li, Y., Cao, L., Xie, X., Zhang, X., Li, Y.: Production optimization of polymer flooding using improved monte carlo gradient approximation algorithm with constraints. Journal of Circuits, Systems and Computers **27**(11), 1850167 (2018). https://doi.org/10.1142/S0218126618501670

[9] Zhou, K., Hou, J., Zhang, X., Du, Q., Kang, X., Jiang, S.: Optimal control of polymer flooding based on simultaneous perturbation stochastic approximation method guided by finite difference gradient. Computers & chemical engineering **55**, 40–49 (2013). https://doi.org/10.1016/j.compchemeng.2013.04.009

[10] Benner, P., Ohlberger, M., Patera, A., Rozza, G., Urban, K. (eds.): Model Reduction of Parametrized Systems. MS&A. Modeling, Simulation and Applications, vol. 17, p. 504. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58786-8. Selected papers from the 3rd MoRePaS Conference held at the International School for Advanced Studies (SISSA), Trieste, October 13–16, 2015

[11] Zahr, M.J., Farhat, C.: Progressive construction of a parametric reduced-order model for PDE-constrained optimization. Int. J. Numer. Meth. Engng **102**, 1111–1135 (2015). https://doi.org/10.1002/nme.4770

[12] Keil, T., Mechelli, L., Ohlberger, M., Schindler, F., Volkwein, S.: A non-conforming dual approach for adaptive trust-region reduced basis approximation of PDE-constrained parameter optimization. ESAIM Math. Model. Numer. Anal. **55**(3), 1239–1269 (2021). https://doi.org/10.1051/m2an/2021019

[13] Banholzer, S., Keil, T., Mechelli, L., Ohlberger, M., Schindler, F., Volkwein, S.: An adaptive projected Newton non-conforming dual approach for trust-region reduced basis approximation of PDE-constrained parameter optimization. arXiv-eprint:2012.11653 (2020)

[14] Gavrilenko, P., Haasdonk, B., Iliev, O., Ohlberger, M., Schindler, F., Toktaliev, P., Wenzel, T., Youssef, M.: A full order, reduced order and machine learning model pipeline for efficient prediction of reactive flows (2021) https://arxiv.org/abs/http://arxiv.org/abs/2104.02800v2. arXiv-eprint:2104.02800

[15] Haasdonk, B., Ohlberger, M., Schindler, F.: An adaptive model hierarchy for data-augmented training of kernel models for reactive flow (2021) https://arxiv.org/abs/http://arxiv.org/abs/2110.12388v1. arXiv-eprint:2110.12388

[16] Saberi, H., Esmaeilnezhad, E., Choi, H.J.: Artificial neural network to forecast enhanced oil recovery using hydrolyzed polyacrylamide in sandstone and carbonate reservoirs. Polymers **13**(16) (2021). https://doi.org/10.3390/polym13162606

[17] Cheraghi, Y., Kord, S., Mashayekhizadeh, V.: Application of machine learning techniques for selecting the most suitable enhanced oil recovery method; challenges and opportunities. Journal of Petroleum Science and

Engineering **205**, 108761 (2021). https://doi.org/10.1016/j.petrol.2021.108761

[18] Ahmadi, M.A.: Developing a robust surrogate model of chemical flooding based on the artificial neural network for enhanced oil recovery implications. Mathematical Problems in Engineering **2015**, 9 (2015). https://doi.org/10.1155/2015/706897

[19] Golzari, A., Haghighat Sefat, M., Jamshidi, S.: Development of an adaptive surrogate model for production optimization. Journal of Petroleum Science and Engineering **133**, 677–688 (2015). https://doi.org/10.1016/j.petrol.2015.07.012

[20] Lee, J.-Y., Shin, H.-J., Lim, J.-S.: Selection and evaluation of enhanced oil recovery method using artificial neural network. Geosystem Engineering **14**, 157–164 (2011). https://doi.org/10.1080/12269328.2011.10541345

[21] Lye, K.O., Mishra, S., Ray, D., Chandrashekar, P.: Iterative surrogate model optimization (ismo): An active learning algorithm for pde constrained optimization with deep neural networks. Computer Methods in Applied Mechanics and Engineering **374**, 113575 (2021). https://doi.org/10.1016/j.cma.2020.113575

[22] Wang, S., Bhouri, M.A., Perdikaris, P.: Fast PDE-constrained optimization via self-supervised operator learning. arXiv-eprint:2110.13297 (2021)

[23] Sarma, P., Durlofsky, L.J., Aziz, K., Chen, W.H.: Efficient real-time reservoir management using adjoint-based optimal control and model updating. Computational Geosciences **10**(1), 3–36 (2006). https://doi.org/10.1007/s10596-005-9009-z

[24] Rasmussen, A.F., Sandve, T.H., Bao, K., Lauser, A., Hove, J., Skaflestad, B., Klöfkorn, R., Blatt, M., Rustad, A.B., Sævareid, O., *et al.*: The open porous media flow reservoir simulator. Computers & Mathematics with Applications **81**, 159–185 (2021). https://doi.org/10.1016/j.camwa.2020.05.014

[25] Bao, K., Lie, K.-A., Møyner, O., Liu, M.: Fully implicit simulation of polymer flooding with mrst. Computational Geosciences **21**(5), 1219–1244 (2017). https://doi.org/10.1007/s10596-017-9624-5

[26] Holmes, J., Barkve, T., Lund, O.: Application of a multisegment well model to simulate flow in advanced wells. In: European Petroleum Conference (1998). https://doi.org/10.2118/50646-MS. OnePetro

[27] Holmes, J.: Enhancements to the strongly coupled, fully implicit well

model: wellbore crossflow modeling and collective well control. In: SPE Reservoir Simulation Symposium (1983). https://doi.org/10.2118/12259-MS. OnePetro

[28] Chen, Z.: Reservoir Simulation: Mathematical Techniques in Oil Recovery. SIAM, Philadelphia (2007)

[29] Lu, R., Reynolds, A.: Joint optimization of well locations, types, drilling order, and controls given a set of potential drilling paths. SPE Journal **25**(03), 1285–1306 (2020). https://doi.org/10.2118/193885-MS

[30] Chen, Y., Oliver, D.S., Zhang, D.: Efficient ensemble-based closed-loop production optimization. SPE Journal **14**(04), 634–645 (2009). https://doi.org/10.2118/112873-PA

[31] Stordal, A.S., Szklarz, S.P., Leeuwenburgh, O.: A theoretical look at ensemble-based optimization in reservoir management. Mathematical Geosciences **48**(4), 399–417 (2016). https://doi.org/10.1007/s11004-015-9598-6

[32] Nocedal, J., Wright, S.: Numerical Optimization. Springer, New York (2006). https://doi.org/10.1007/978-0-387-40065-5

[33] Montgomery, D.C., Jennings, C.L., Kulahci, M.: Introduction to Time Series Analysis and Forecasting. John Wiley & Sons, New Jersey (2015)

[34] Fonseca, R.R.-M., Chen, B., Jansen, J.D., Reynolds, A.: A stochastic simplex approximate gradient (stosag) for optimization under uncertainty. International Journal for Numerical Methods in Engineering **109**(13), 1756–1776 (2017). https://doi.org/10.1002/nme.5342

[35] Oguntola, M.B., Lorentzen, R.J.: Ensemble-based constrained optimization using an exterior penalty method. Journal of Petroleum Science and Engineering **207**, 109165 (2021). https://doi.org/10.1016/j.petrol.2021.109165

[36] Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. 10.1007/978-0-387-84858-7, New York (2009). https://doi.org/10.1007/978-0-387-84858-7

[37] Petersen, P., Voigtlaender, F.: Optimal approximation of piecewise smooth functions using deep relu neural networks. Neural Networks **108**, 296–330 (2018). https://doi.org/10.1016/j.neunet.2018.08.019

[38] Elbrächter, D., Grohs, P., Jentzen, A., Schwab, C.: Dnn expression rate analysis of high-dimensional pdes: Application to option pricing. Constructive Approximation (2021). https://doi.org/10.1007/

s00365-021-09541-6

[39] Rumelhart, D.E., Hintont, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature **323**(6088), 533–536 (1986). https://doi.org/10.1038/323533a0

[40] Bottou, L., Curtis, F.E., Nocedal, J.: Optimization methods for large-scale machine learning. SIAM Rev. **60**(2), 223–311 (2018). https://doi.org/10.1137/16M1080173

[41] Liu, D.C., Nocedal, J.: On the limited memory BFGS method for large scale optimization. Mathematical Programming **45**, 503–528 (1989). https://doi.org/10.1007/BF01589116

[42] Prechelt, L.: Early stopping - but when? In: Neural Networks: Tricks of the Trade, Volume 1524 of LNCS, Chapter 2, pp. 55–69. Springer, Berlin, Heidelberg (1997). https://doi.org/10.1007/978-3-642-35289-8_5

[43] He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: 2015 IEEE International Conference on Computer Vision (ICCV), pp. 1026–1034 (2015). https://doi.org/10.1109/ICCV.2015.123

[44] Yarotsky, D.: Error bounds for approximations with deep ReLU networks. Neural Networks **94**, 103–114 (2017). https://doi.org/10.1016/j.neunet.2017.07.002

[45] LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**, 436–44 (2015). https://doi.org/10.1038/nature14539

[46] Fonseca, R., Kahrobaei, S., Van Gastel, L., Leeuwenburgh, O., Jansen, J.: Quantification of the impact of ensemble size on the quality of an ensemble gradient using principles of hypothesis testing. In: SPE Reservoir Simulation Symposium (2015). https://doi.org/SPE-173236-MS. OnePetro

[47] Baxendale, D., Rasmussen, A.F., Rustad, A.B., Skille, T., Sandve, T.H.: Opm flow documentation manual. Open Porous Media Initiative (2021)

[48] Milk, R., Rave, S., Schindler, F.: pyMOR – generic algorithms and interfaces for model order reduction. SIAM J. Sci. Comput. **38**(5), 194–216 (2016). https://doi.org/10.1137/15m1026614

[49] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Wallach, H., Larochelle, H.,

Beygelzimer, A., d' Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc., Vancouver, BC, Canada (2019)

[50] Wolfe, P.: Convergence conditions for ascent methods. SIAM Review **11**(2), 226–235 (1969). https://doi.org/10.1137/1011036

[51] Wolfe, P.: Convergence conditions for ascent methods. ii: Some corrections. SIAM Review **13**(2), 185–188 (1971). https://doi.org/10.1137/1013035

[52] Hofmann, T., Schölkopf, B., Smola, A.J.: Kernel methods in machine learning. The Annals of Statistics **36**(3), 1171–1220 (2008). https://doi.org/10.1214/009053607000000677