Master's thesis

Arpita Chaturvedi

Well Control Optimization by Coupling Smart Proxy Models with Genetic Algorithm

Master's thesis in Petroleum Engineering
Supervisor: Ashkan Jahanbani Ghahfarokhi
Co-supervisor: Mathias Bellout
June 2021

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Geoscience and Petroleum

NTNU
Norwegian University of Science and Technology

Arpita Chaturvedi

Well Control Optimization by Coupling Smart Proxy Models with Genetic Algorithm

Master's thesis in Petroleum Engineering
Supervisor: Ashkan Jahanbani Ghahfarokhi
Co-supervisor: Mathias Bellout
June 2021

Norwegian University of Science and Technology
Faculty of Engineering
Department of Geoscience and Petroleum

NTNU
Norwegian University of
Science and Technology

Acknowledgement

This thesis is submitted in partial fulfilment of the MSc degree requirement in Reservoir Engineering and Petrophysics at the Department of Geoscience and Petroleum at Norwegian University of Science and Technology (NTNU). In addition, this study was done in close collaboration with a BRU21 PhD project by Semyon Fedorov at Department of Industrial Economics and Technology Management (NTNU) and OKEA AS.

The work is intended for everyone interested in applying artificial intelligence in the petroleum industry, especially field development optimization. Basic knowledge of reservoir simulation and optimization algorithms are recommended but not required to understand the fundamental ideas presented in this work.

First and foremost, I would like to thank my supervisor, Professor Ashkan Jahanbani, and co-supervisor Mathias Bellout. The technical inputs and the resources I got to know through them have been invaluable for this thesis. I wish to express my sincere gratitude to Ashkan for introducing me to this topic, giving me excellent guidance and supporting me throughout the thesis. Ashkan has provided valuable insight into the topic of smart proxy modeling and has been incredible in terms of discussing and giving me the freedom to learn and solve problems. It is such an honor being supervised and working with him. I would also like to thank my co-supervisor Mathias Bellout, for his meaningful suggestions.

Furthermore, I would like to thank postdoctoral fellow Nait Amar Menad and PhD candidate Cuthbert Shang Wui Ng for providing continuous feedback during bi-weekly meetings and directing me to the right literature, as well as for the invaluable discussions and suggestions to improve my work. I am incredibly grateful for all the technical insights and conversations I had with Cuthbert.

A big thanks to my parents for supporting me to continue my education and being a pillar of support throughout my life. A huge appreciation also goes to my boyfriend, Gurveer Singh, for keeping me motivated, and for always looking at the brighter side of life.

A.C.

Abstract

Generally, optimal well controls to maximize net present value (NPV) are obtained by coupling of numerical reservoir simulation with an optimization algorithm. This approach requires a significant number of numerical reservoir simulations that are computationally expensive and time-consuming due to complex flow behavior of reservoir. As a result, it becomes a necessity to develop a fast and accurate alternative. Light mathematical models, such as proxy models, have a high capability to identify very complex and non-linear behavior in short time, such as complex dynamic flow behavior of reservoir.

This study proposes a methodology that begins by developing smart proxy models (SPMs) for a synthetic field model, based on Artificial Neural Networks (ANNs) and then integrate the established proxy models with Genetic Algorithm (GA) to solve the well control optimization problem. Three ANN models are developed using reservoir simulator data to predict field production profiles, i.e., field oil production rate, field water injection rate, and field water production rate based on sets of well control values, i.e., bottom-hole pressures (BHP). Latin hypercube sampling is used to prepare the database utilized for constructing SPMs. Hyperparameter optimization study assists in finding the best ANN architecture for each proxy. Various performance metrics are explored to comment on "goodness" of the proxy models. The proposed methodology also includes sensitivity study of GA control parameters using SPM-GA coupling and introduces the possibility for occasional retraining and multiple quality checks of ANNs as more data is gathered. From SPM-GA coupling, the optimum well control parameters, namely bottom hole pressures of injectors and producers, which maximize net present value, are investigated.

The developed proxy models produce outputs within seconds, while the reservoir simulator takes an average time of 30 minutes for the synthetic field i.e. Olympus. SPM-GA coupling works well for well control optimization study by finding BHP configuration that gives significant increase of 35% in net present values (NPV), and requires fewer numerical simulations compared to the traditional approach. The results show that the established proxy models are found to be robust and efficient tools for mimicking the numerical simulator performance in well control optimization. Significant reduction in computational time and resources is observed.

Table of Contents

List of Tables

List of Figures

Abbreviations

AI	-	Artificial Intelligence
ANN	-	Artificial Neural Network
API	-	Application Programming Interface
APPS	-	Asynchronous Parallel Pattern Search
BHP	-	Bottom Hole Pressure
CMA-ES	-	Covariance Matrix Adaptation Evolution Strategy
DOE	-	Design of Experiments
GA	-	Genetic Algorithm
GPS	-	Generalized Pattern Search
HPO	-	Hyperparameter Optimization
FOPR	-	Field Oil Production Rate
FOPT	-	Field Oil Production Total
FWIR	-	Field Oil Injection Rate
FWIT	-	Field Oil Injection Total
LHS	-	Latin Hypercube Sampling
LSTM	-	Long Short Term Memory
MAE	-	Mean Absolute Error
MAPE	-	Mean Absolute Percentage Error
ML	-	Machine Learning
MMN	-	Min-Max Normalization
MSE	-	Mean Squared Error
NCS	-	Norwegian Continental Shelf
NPV	-	Net Present Value
NRMSE	-	Normalized Root Mean Squared Error
NTG	-	Net to Gross
OWC	-	Oil Water Contact
PERMX	-	Permeability in X Direction
PERMY	-	Permeability in Y Direction
PERMZ	-	Permeability in Z Direction
PORO	-	Porosity
PPS	-	Parallel Pattern Search
PSO	-	Particle Swarm Optimization
ReLU	-	Rectified Linear Unit
RSM	-	Response Surface Methodology
RMSE	-	Root Mean Squared Error
RWS	-	Roulette Wheel Selection
SD	-	Standard Deviation
SPM	-	Smart Proxy Model
SRM	-	Surrogate Reservoir Model
USD	-	United States Dollar

Chapter 1

Introduction

This opening chapter presents brief explanation of the motivation behind this master's thesis and its objectives.

1.1 Background

The first phase in oil and gas development is primary recovery with the support of natural drive mechanism present in the reservoir. Reservoir's natural driving mechanism becomes incapable of supporting an efficient and economically attractive oil recovery, as the pressure in the reservoir decreases due to oil production; thus, a secondary recovery method is used. The most widely used secondary recovery method to increase oil production and ultimate hydrocarbon recovery is waterflooding. Injected water helps to retain the pressure in the long run and displaces the oil towards the producers, hence, resulting in an improved sweep efficiency. Optimization studies are conducted to find the optimal well control settings for producers and injectors that maximize net present value (NPV) or the ultimate hydrocarbon recovery. In general, coupling numerical simulations with a suitable optimization algorithm searches a much larger problem space compared to experimental studies carried out in reservoir labs (Ma et al., 2019). Optimization methods are essential for determining the best field development strategy. Well placement/selection, well control, and completion design optimization are examples of optimization problems encountered during the field development process. In recent years, optimization algorithms such as evolutionary strategies and gradient-based techniques have been widely applied to the field development optimization problems (Isebor et al., 2014; Bellout and Volkov, 2018). However, application of these optimization algorithms requires a significant number of computationally expensive dynamic simulations to obtain optimal solutions. Moreover, stochastic optimization algorithms due to population based nature, require even a larger number of function evaluations. This makes direct application of these optimization algorithms with a numerical simulator a time-consuming and expensive procedure.

As discussed in (Khor et al., 2017), various techniques with varying computational requirements have been proposed to address the problems encountered in the oil field development and production system optimization. Techniques implemented to reduce the computational cost associated with optimization problems can be divided into two general approaches. The first approach addresses flow problems by replacing complex numerical simulation models with reduced-order models or models based on simplified flow physics. In contrast, second set of techniques entails reducing the dimension of the optimization problem, i.e. reducing the number of optimization variables. Methods based on the first approach include employing reduced-order numerical models (Van Doren et al., 2006; Jansen and Durlofsky, 2017) and virtual intelligence-based proxies (Ma et al., 2019; Nasir, 2020; Jin et al., 2020; Tang et al., 2021). The second approach includes methods such as multilevel optimization (Awotunde and Sibaweihi, 2018) and two-stage optimization strategy (Nasir et al., 2021), to name a few. Awotunde (2019) provides a review of several dimension reduction techniques for well control optimization. This thesis uses the first approach and proposes implementing smart proxy models (SPMs) based on artificial intelligence (AI) techniques to represent the non-linear dynamic behavior of the reservoir.

AI-based techniques have been applied in petroleum and reservoir engineering to solve a variety of conventional and unconventional problems (Ridha and Mansoori, 2005; Mohaghegh, 2007; Nait Amar et al., 2020). Among AI methods, artificial neural network (ANN) is the most well-known due to its effectiveness and flexibility in recognizing highly complex and non-linear systems (as in the case of numerical simulator's responses) within a short time frame (Nait Amar et al., 2018). Some areas with successful applications of ANNs as proxy models in reservoir engineering, include optimization studies (Sayyafzadeh and Alrashdi, 2019; Nait Amar et al., 2020; Nasir, 2020; Teixeira and Secchi, 2019), history matching (Shahkarami et al., 2014), uncertainty studies (Mohaghegh et al., 2006). Since the output of reservoir forecasting is a sequential problem, researchers have begun to use the long short-term memory (LSTM) network algorithm in recent years. However, oil production data is strongly phased in nature and is often divided into periods of increasing production, stable production and decreasing production. Tang et al. (2021) stated that when using standard LSTM neural networks to forecast oil field productions, there will be issues with neural network generalization, low prediction precision, and even negative values for the expected output, with significant deviations (Peng et al., 2020).

To overcome the shortcomings of existing solutions, ANN is used in this work to build multiinput and single output proxy models to predict dynamic field productions. In contrast to the existing research, influence of inputs from previous timesteps on current timestep is not emphasized; instead, this behavior is left to the ANNs to learn. These models are built using data generated from a commercial numerical reservoir simulator, named Eclipse. Proposed models are coupled with an appropriate optimization algorithm to determine the optimal well control settings that give the highest NPV. SPM-GA coupling has a faster optimization speed, more comparison schemes, and can find the optimum control frequency.

Nasir (2020) stated that field development optimization problem is often solved separately for each petroleum field due to variation in geological models and constraints involved, which further results in a large number of costly flow simulation runs for each field under consideration. This suggests that field development optimization can benefit from methodologies with a generalized optimization workflow that can be applied to several petroleum fields. Therefore, this thesis proposes a workflow that allows for multiple quality checks and occasional retraining of the SPMs. Furthermore, choice of performance metrics to evaluate the accuracy of the SPMs are examined, in addition to the usefulness of the established SPMs to solve the well control optimization. This thesis also provides a comprehensive review of the literature on the ANN models utilized in this study, so that individuals with no prior knowledge of the subject can also understand the foundation of the thesis.

1.2 Objectives

The thesis's main objective is to develop a methodology for constructing SPMs and integrate SPMs with an optimization algorithm to solve well control optimization task. This methodology is implemented based on a synthetic reservoir model, named Olympus.

Following tasks are performed to achieve the main objective of this thesis:

1. Define and formulate optimization problem to be solved.
2. Generate simulation cases using an appropriate sampling method in defined solution space.
3. Build dataset consisting of necessary information extracted from the reservoir simulator and then divide it into training, validation, and test samples. Generate more dataset for blind testing.

4. Construct SPMs to predict field oil production rate, field water production rate and field water injection rate.

5. Evaluate performance of proxies using suitable accuracy metrics.

6. Set a base case well control values.

7. Use SPM-GA coupling for sensitivity analysis of GA control parameters

8. Quality check and retrain proxy models as more data is gathered.

9. Find the optimal BHPs configuration corresponding to maximum NPV obtained from SPM-GA coupling

10. Simulate a case with the optimal BHPs control settings obtained from SPM-GA coupling using the numerical reservoir simulator and calculate the NPV.

11. Compare the optimum NPV with the base case and analyze the results.

1.3 Outline

The rest of the thesis contains the following sections in chronological order:

- Chapter 2. Theory: presents theories and principles used in this study, including literature studies and how they are relevant.

- Chapter 3. Methodology: presents project workflow showing how the tasks were approached to reach the main objective and the setup for optimization problem solved in this study.

- Chapter 4. Results: presents findings and outcomes of the tasks involved in development of SPMs and results of optimization runs.

- Chapter 5. Discussion: discusses implications of results obtained in Chapter 4, in addition to limitations of the work.

- Chapter 6. Conclusion and Recommendation for further work: this chapter sums up the most important findings based on the results obtained, and provides ideas for further research.

Chapter 2

Theory

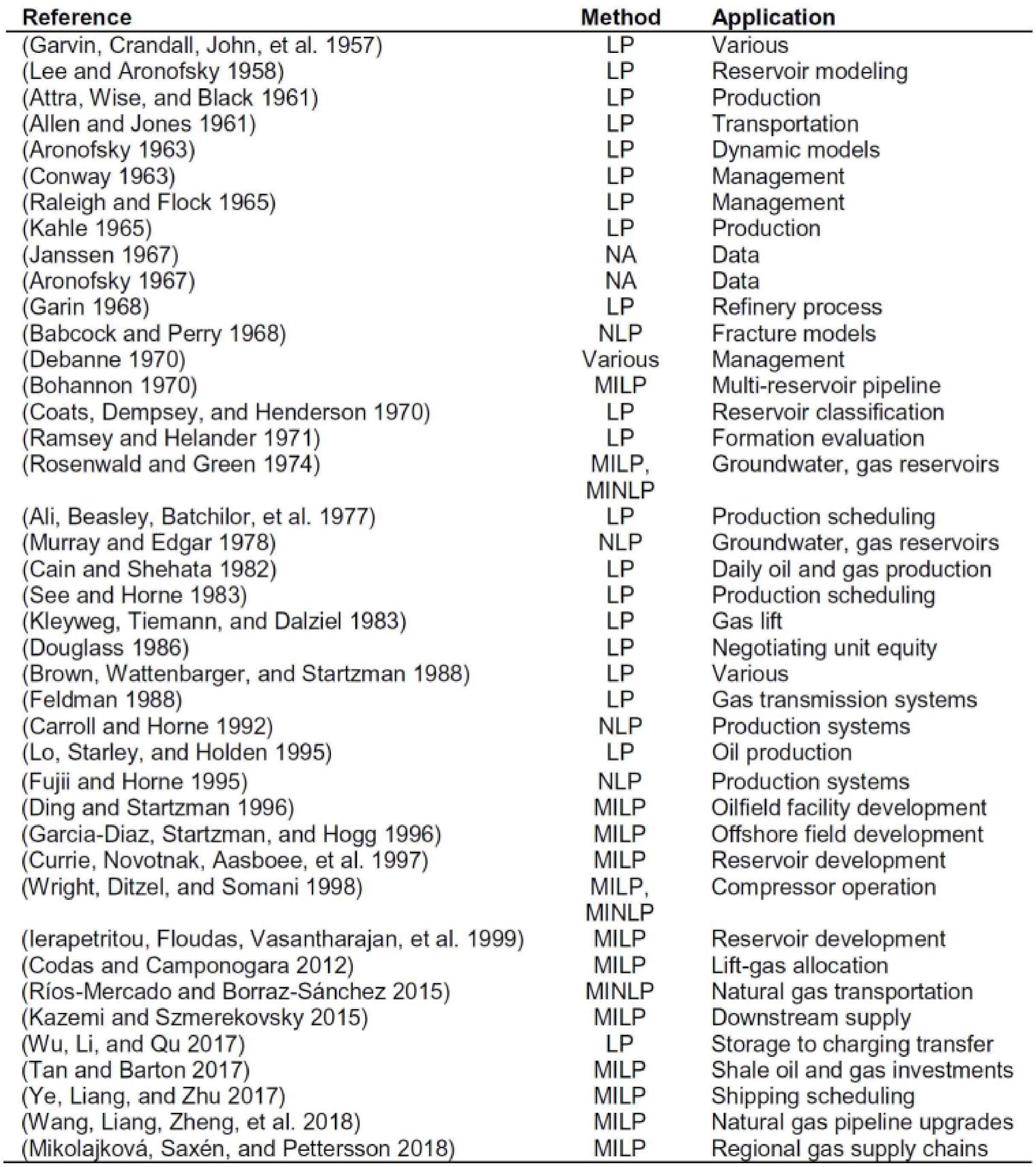
This chapter presents underlying theories and principles used in this study, including literature reviews and their relevance to this study. This chapter covers the aspects of field development optimization, optimization algorithms, and artificial intelligence.

2.1 Field Development Optimization[1]

Petroleum field development encompasses operations involving various engineering disciplines. A range of decisions can be made for various aspects of field development such as drilling, facility operation, and reservoir production by solving an associated optimization problem. Field development decisions with reservoir related focus deal with the way wells are configured within a reservoir (Baumann et al., 2020). Potential optimization parameters can be well configuration parameters, e.g., number, type, location, completion design and control settings of injection/production wells. The well-level optimization problem can be divided into three categories: well control, well completion design and well placement. This work focuses only on well control optimization that is explained below. However, well placement optimization is also discussed to develop an understanding of the topic. Figure 2.1 [2] summarizes papers from a historical study, along with their focus area and optimization methods applied in the petroleum industry (DiCarlo et al., 2019).

[1]This section is adapted and modified from author's specialization project (Chaturvedi, 2020).

[2]In Fig. 2.1, LP is an abbreviation for Linear Programming, NLP for Non-Linear Programming, MILP for Mixed-Integer Linear Programming, and MINLP for Mixed-Integer Non-Linear Programming.



Reference	Method	Application
(Garvin, Crandall, John, et al. 1957)	LP	Various
(Lee and Aronofsky 1958)	LP	Reservoir modeling
(Attra, Wise, and Black 1961)	LP	Production
(Allen and Jones 1961)	LP	Transportation
(Aronofsky 1963)	LP	Dynamic models
(Conway 1963)	LP	Management
(Raleigh and Flock 1965)	LP	Management
(Kahle 1965)	LP	Production
(Janssen 1967)	NA	Data
(Aronofsky 1967)	NA	Data
(Garin 1968)	LP	Refinery process
(Babcock and Perry 1968)	NLP	Fracture models
(Debanne 1970)	Various	Management
(Bohannon 1970)	MILP	Multi-reservoir pipeline
(Coats, Dempsey, and Henderson 1970)	LP	Reservoir classification
(Ramsey and Helander 1971)	LP	Formation evaluation
(Rosenwald and Green 1974)	MILP, MINLP	Groundwater, gas reservoirs
(Ali, Beasley, Batchilor, et al. 1977)	LP	Production scheduling
(Murray and Edgar 1978)	NLP	Groundwater, gas reservoirs
(Cain and Shehata 1982)	LP	Daily oil and gas production
(See and Horne 1983)	LP	Production scheduling
(Kleyweg, Tiemann, and Dalziel 1983)	LP	Gas lift
(Douglass 1986)	LP	Negotiating unit equity
(Brown, Wattenbarger, and Startzman 1988)	LP	Various
(Feldman 1988)	LP	Gas transmission systems
(Carroll and Horne 1992)	NLP	Production systems
(Lo, Starley, and Holden 1995)	LP	Oil production
(Fujii and Horne 1995)	NLP	Production systems
(Ding and Startzman 1996)	MILP	Oilfield facility development
(Garcia-Diaz, Startzman, and Hogg 1996)	MILP	Offshore field development
(Currie, Novotnak, Aasboee, et al. 1997)	MILP	Reservoir development
(Wright, Ditzel, and Somani 1998)	MILP, MINLP	Compressor operation
(Ierapetritou, Floudas, Vasantharajan, et al. 1999)	MILP	Reservoir development
(Codas and Camponogara 2012)	MILP	Lift-gas allocation
(Ríos-Mercado and Borraz-Sánchez 2015)	MINLP	Natural gas transportation
(Kazemi and Szmerekovsky 2015)	MILP	Downstream supply
(Wu, Li, and Qu 2017)	LP	Storage to charging transfer
(Tan and Barton 2017)	MILP	Shale oil and gas investments
(Ye, Liang, and Zhu 2017)	MILP	Shipping scheduling
(Wang, Liang, Zheng, et al. 2018)	MILP	Natural gas pipeline upgrades
(Mikolajková, Saxén, and Pettersson 2018)	MILP	Regional gas supply chains

Figure 2.1: Summary of optimization methods and applications in petroleum industry (DiCarlo et al., 2019).

2.1.1 Well Control Optimization

Well control optimization deals with finding the optimum settings of control parameters that, for instance, maximize oil and gas production and decrease water production or optimize injection schedules, to improve sweep efficiency (water flooding). Commonly used control variables are well bottom hole pressures (BHP), rates, or a combination of BHPs and rates, for producers and injectors at a specific time.

Many methodologies developed to solve well control optimization problem employ a variety of optimization algorithms. Wang et al. (2019) used both deterministic (generalized pattern search (GPS)) and stochastic algorithms (particle swarm optimization (PSO) and covariance matrix adaptation evolution strategy (CMA-ES)) together with a multiscale approach with variable control frequencies to solve the problem. (Isebor et al., 2014) used derivative-free, and (Arouri and Sayyafzadeh, 2020) used gradient based methods. In the traditional approach, the BHPs or rates are passed to a reservoir simulator such as Eclipse to obtain values of different components required for NPV computation. In contrast, this thesis encourages use of proxy models for optimization study by passing control variables as inputs into the proxy models and obtaining desired outputs.

2.1.2 Well Placement Optimization

Goal of well placement optimization is to find optimal locations to drill a new producer or injector. This is an essential task because it determines the area of a reservoir that can be produced or flooded. The reservoir management team specifies the total number and type of wells and their operational settings, such as BHP or injection rates, to be used in determining optimal well locations. Constraints are typically imposed to ensure well placement within a specific region of reservoir based on engineering experiences and specific reservoir knowledge such as faults and thief zones; it is critical to impose limitations on the feasibility search area of new wells (Sadigov, 2019). It is necessary to accurately translate these identified constraints into reasonable constraints within the problem formulation process. Once an optimization framework is set up, the success of optimization work depends on both the optimizer's constraint handling capability and the efficiency of the search algorithm being implemented (Jesmani et al., 2015). The constraints ensure that the optimizer search for the optimal solution in the specified region of formation, maintain inter-well distance and avoid drilling wells in challenging geological structures. Similar to well control, many methodologies are developed and applied to well placement optimization problems as well, including derivative-free (Badru and Kabir, 2003; Jesmani et al., 2015; Maschio et al., 2008; Sadigov, 2019) and gradient-based methods (Volkov and Bellout, 2018).

2.2 Optimization Theory

2.2.1 Optimization Problem[3]

This section discusses the optimization problem in general and specifically describes the well control optimization problem. An optimization problem consists of three components: the objective function, control variables, and constraints. The mathematical formulation of the optimization problem can be written as follows (Nocedal and Wright, 2006) :

$$min_{x\in\mathbb{R}^n} f(x) \quad \text{subject to } c_i(x) = \begin{cases} 0, & i \in E \\ \geq 0, & i \in I \end{cases}$$

Here,

- x is the vector of variables, also called *unknowns* or *parameters*,
- f is the *objective function* we want to minimize or maximize,
- c_i is *constraint function*, that is scalar function of x which defines certain equations and inequalities that the vector x must satisfy,
- E and I are sets of indices for equality and inequality constraints, respectively.

In field development optimization problems, a reservoir simulator is used to solve fluid flow equations in porous media to compute the objective function associated with a given input variable. The *objective function* in such problems is usually NPV or the weighted sum of cumulative fluid productions from the reservoir.

The typical objective function associated with well control optimization problem evaluates NPV while accounting for various costs such as oil price, injection cost, and water production cost. In this work, the objective function of interest is the NPV, and the author only deals with bound constraints. Equation 2.1 is one method for calculating NPV (Bellout, 2014):

$$NPV(x,u) = \sum_{k=1}^{N_s} \left(\sum_{j=1}^{N_p} p_o {q_o}^{j,k}(u,x)\Delta t_k - \sum_{j=1}^{N_p} c_{wp} {q_{wp}}^{j,k}(u,x)\Delta t_k - \sum_{j=1}^{N_i} c_{wi} {q_{wi}}^{j,k}(u,x)\Delta t_k \right) \Big/ (1+d)^t \tag{2.1}$$

[3]This section is adapted and modified from author's specialization project (Chaturvedi, 2020).

Where N_p and N_i are total number of producers and injectors in the system, $q_o^{j,k}$,$q_{wp}^{j,k}$, and $q_{wi}^{j,k}$ are the flow rates of the oil, water produced and water injected for well j at the output interval k, respectively, and Δt_k is the length of each N_s time steps of simulation. The oil price, the cost of water produced and injected are represented by p_o, c_{wp} and c_{wi}, respectively. d stands for discount rate expressed in fraction, and t is the total number of years, starting from zero at first year. Discount rate is defined as interest rate that is used for calculating the present value of future cash flow of a project (Chang et al., 2019). The author uses 0.08 discount factor in this work.

2.2.2 Optimization Algorithms

A mathematical optimization is defining an objective function and then finding an input value within a predefined space corresponding to maximum or minimum of the objective function. An optimization algorithm is a procedure that is iteratively executed by comparing various solutions until an optimum solution is found. Depending on the focus or the characteristics used for comparison, optimization algorithms can be classified in various ways. One approach to categorize optimization algorithms is based on derivative information of objective function, which separates these algorithms into gradient-based (or derivative-based) and gradient-free (or derivative-free). Another way to classify optimization algorithms is by search strategy, which might be deterministic or stochastic. Deterministic and stochastic algorithms are two types of widely used optimization algorithms. A deterministic algorithm will always produce the same output when given a particular set of input because the solution of this method depends on the initial seeds. On the contrary, stochastic optimization consists of algorithms that solve an optimization problem by including mathematical randomness in their search strategy (Cavazzuti, 2013). This project is mainly concerned with genetic algorithm (GA) that is stochastic and population based. GA, PSO and APPS are discussed in detail further.

Genetic Algorithm (GA)

John Holland along with his collaborators proposed this algorithm based on Darwin's theory of evolution (Holland, 1975). GA is a population-based stochastic and derivative-free optimization algorithm. GA often employs biological concepts such as selection, inheritance, mutation, crossover, parents, children, offspring, and reproduction. An encoded string representing an individual solution is referred to as a *chromosome*, and the values of objective functions are referred to as *fitness*. This algorithm is better suited if the objective function is non-smooth, time-consuming to evaluate, or noisy in some way.

Each individual's values are then ranked from best to worst. The selection, crossover, and mutation process are then applied. This results in a new generation whose properties are selected from high-rank generation through selection and crossover. In addition, mutation helps in preventing the solution from converging to local optima. The process is then iterated until the fittest individual is identified, and each iteration results in a new generation. The algorithm will stop when it exceeds the targeted fitness level or the maximum number of generations. The final individual with the best solution in the solution space is selected as the optimization problem solution. Figure 2.2 illustrates the typical GA workflow followed by description of the steps involved.

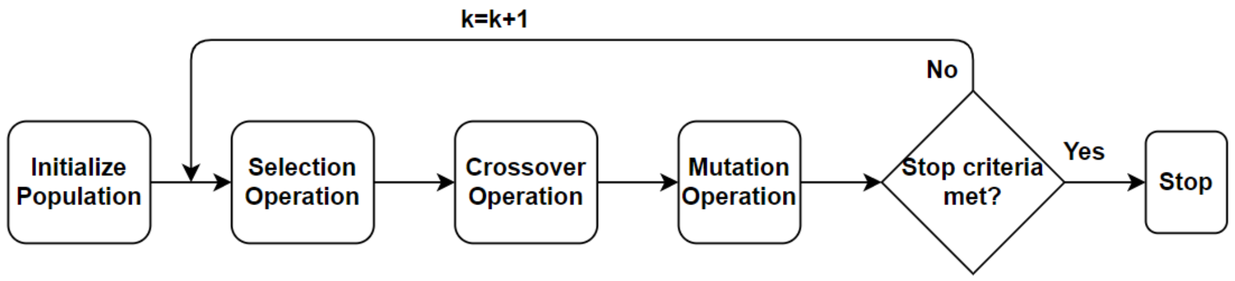


Figure 2.2: Flowchart showing GA optimization scheme, modified from (Chuang et al., 2016).

Initialization

Evolution usually begins with a **population** of randomly generated individuals. Each individual is the proposed solution to the defined problem and has properties that later can be altered. These properties can be defined in binary, permutation, or real number encoding. Every individual is a set of unique variables known as *gene*. These genes are then linked together to form an individual, also known as a **chromosome**. Figure 2.3 illustrates the terms for an easier comprehension.

Selection Operation

The selection operation is based on the fitness of each individual. The fitness value represents the quality of the solution and is used to select the best individuals. Best individuals are chosen as parents to create a mating pool. Then, they produce offsprings that inherit the parents' characteristics and are passed on to the next generation. The higher the parents' fitness, the better the offspring's fitness, thus, the better the chances of survival. Therefore, individuals with the highest fitness are more likely to get selected to breed a new generation. After calculating the fit-

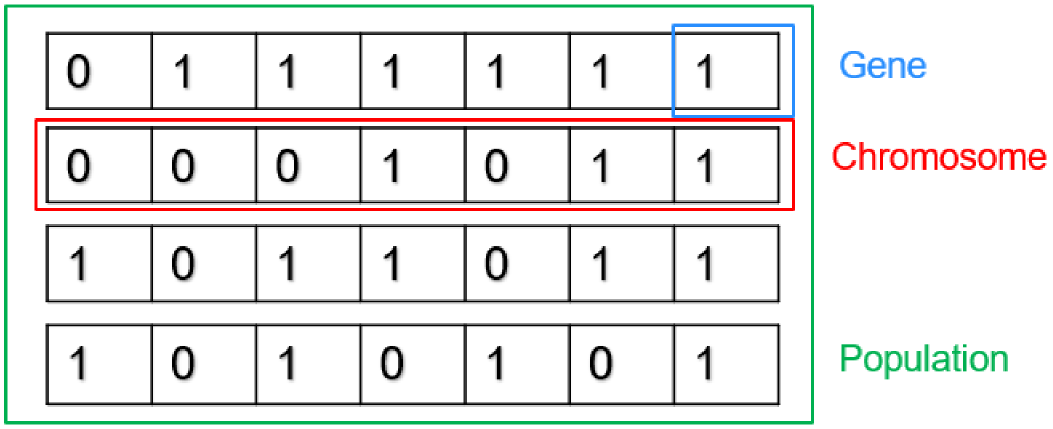


Figure 2.3: Illustration of population, genes, and chromosome in GA.

ness of each individual, a specific selection method is applied. Selection methods are typically probabilistic; roulette wheel and elitism selection are two of selection methods.

Roulette wheel selection (RWS) is closely similar to repeatedly spinning a one-armed roulette wheel, with the sizes of the holes reflecting the selection probabilities (Eiben and Smith, 2003). In the RWS, probability (P_i) of selecting an individual i is calculated by dividing fitness of that individual , f_i by total fitness of the generation with population size of N, as shown in Eq. 2.2 (Grefenstette, 2000).

$$P_i = \frac{f_i}{\sum_{i=1}^{N} fi} \tag{2.2}$$

Elitism selection method transfers a small portion of best individuals in the current population to next generation without any changes (Grefenstette, 2000). This is done to ensure that maximum fitness value within the population is never reduced. These selected individuals are also referred to as **Parents**.

Variation Operator

Specific variation operators, such as mutation and crossover, are applied to the parent chromosomes in each iteration to generate offspring chromosomes. Both variation operators are stochastic, as outputs depend on outcomes of series of random choices (Eiben and Smith, 2003). Crossover is a binary variation operator, and as the name indicates, it merges information from two parents chosen from the pool of selected individuals to produce one or two offsprings. Recombination works based on the principle that by mating two individuals with different yet desirable characteristics, one can produce offspring with each of those characteristics (Eiben and Smith, 2003).

Mutation is a unary variation operator. It is applied to one individual and delivers a slightly modified mutant, the child. It alters one or more genes in a chromosome from its initial value. While most GAs combine mutation and crossover, the mutation is often treated as a background operator to ensure that the population has a diverse pool of individuals that can be manipulated by crossover (Grefenstette, 2000). Therefore, mutation is applied to preserve population diversity, prevent premature convergence, and avoid being stuck in the local optima if it ever becomes trapped.

Termination

The generational process will continue until the termination condition is satisfied. Followings are, but not limited to, some of the termination options for GA (Eiben and Smith, 2003):

- The total number of fitness evaluations reaches a predefined threshold.
- Maximum allowed CPU time elapses.
- The fitness improvement remains below a certain threshold for some time (i.e., for many generations or fitness evaluations).

Particle Swarm Optimization[4]

Kennedy and Eberhart introduced PSO in 1995 (Eberhart and Kennedy, 1995). PSO is a population-based stochastic algorithm that has been widely and efficiently deployed in nonlinear optimizations of varying complexities (Nwankwor et al., 2013). At any given iteration, individuals of the population are referred to as particles representing possible or potential solution to an optimization problem, and collection of particles is called swarm. PSO algorithm solves the problem by generating a swarm of particles representing solution vectors in the search space and updates the particles based on the information obtained from the previous run to find the optimal solution.

In the beginning, each particle is assigned a random velocity and position in search space. At any given iteration, movement of particles is influenced by two factors: cognitive and social

[4]This section is adapted and modified from author's specialization project (Chaturvedi, 2020).

behavior. The cognitive factor allows movement towards local best solution based on iteration-to-iteration information, and the social factor is responsible for attraction towards global best solution based on particle-to-particle interaction. A particle has memory of previous best value of the objective function and corresponding best position vector; this is called *local best* (pbest). In addition, each particle stores in its memory the best solution attained by any particle in the swarms also referred to as *global best* (gbest) and experiences attraction towards this solution. At the end of each iteration, *pbest* and *gbest* are updated for each particle, and this process continues until it reaches stopping criteria. Wang et al. (2019) commented on importance of population size in algorithm's ability to utilize good initial guess. They found that effect of initial guess is important for larger population size. Figure 2.4 illustrates a flowchart for the standard PSO algorithm.

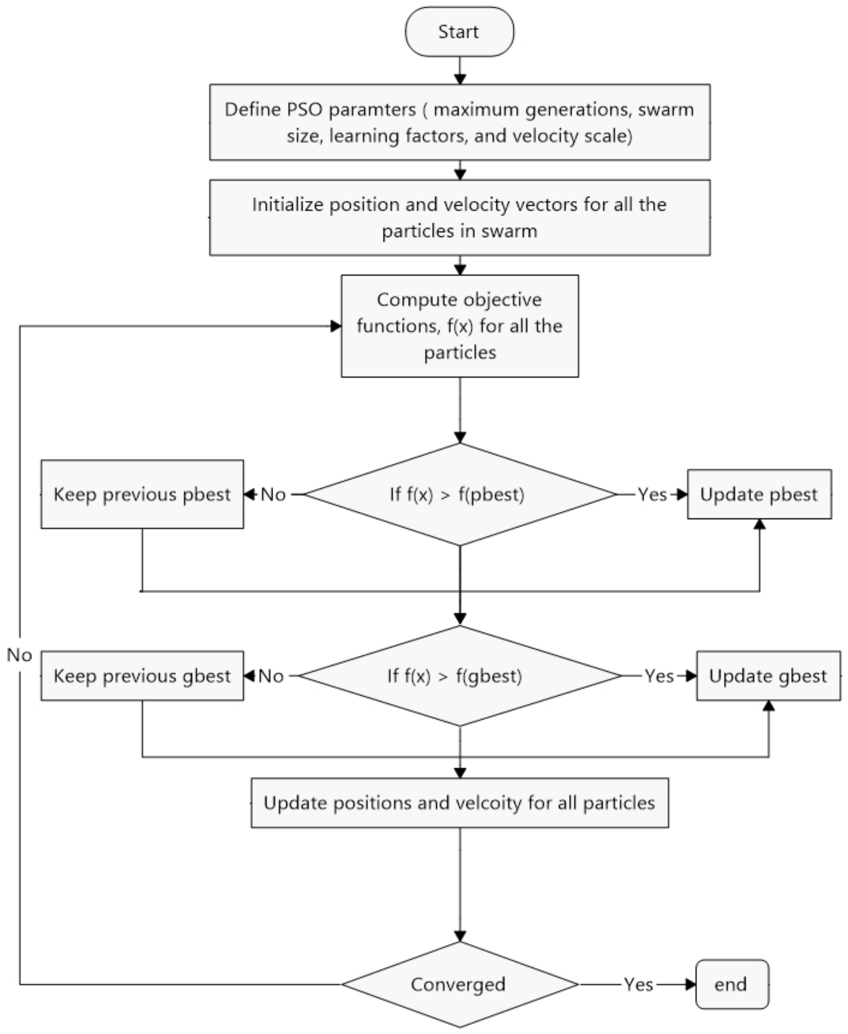

Start
Define PSO paramters (maximum generations, swarm size, learning factors, and velocity scale)
Initialize position and velocity vectors for all the particles in swarm
Compute objective functions, f(x) for all the particles
Keep previous pbest
No
If f(x) > f(pbest)
Yes
Update pbest
No
Keep previous gbest
No
If f(x) > f(gbest)
Yes
Update gbest
Update positions and velcoity for all particles
Converged
Yes
end

Figure 2.4: Flowchart showing PSO algorithm.

At iteration k, if position of i-th particle is represented by a d-dimensional vector $x_i(k)$ and velocity is given by vector $v_i(k)$, local best position of particle i is represented by $y_i(k)$ and global best position attained from particle-to-particle interaction is given by $y^*(k)$, and l_1, l_2 are learning factors responsible for cognitive and social behavior, respectively, position and velocity at next iteration can be computed by Eq. 2.3:

$$x_i(k+1) = x_i(k) + v_i(k+1) \tag{2.3}$$

where, elements of updated velocity vector $v_i(k+1)$ are given by Eq. 2.4

$$v_{i,j}(k+1) = v_{i,j}(k) + l_1 r_{1,j}\left(y_{i,j}(k) - x_{i,j}(k)\right) + l_2 r_{2,j}\left(y^*_{i,j}(k) - x_{i,j}(k)\right) \tag{2.4}$$

Here, $j = 1,2,3...d$ represents the components of dimensions in search space, $r_{1,j}$ and $r_{2,j}$ are random numbers ranging between 0 and 1. Due to its stochastic nature, PSO algorithm does not get trapped in local optimum and is computationally efficient for problems containing a higher number of control variables. The total number of control variables does not have any impact on the number of evaluations. Therefore, PSO is a good choice for optimization problems with many control variables. In general, effectiveness and problem solving by PSO is a population-based phenomenon, emerging from the individual behavior of particles through their interaction with neighbours in the swarm (Nwankwor et al., 2013).

Asynchronous Parallel Pattern Search[5]

APPS algorithm was used to solve the optimization task investigated in the author's specialization project (Chaturvedi, 2020). APPS is a deterministic and derivative free optimization algorithm with the unique feature that dynamically makes decisions without waiting for the information on all processors to be available (Kolda et al., 2003). APPS focuses on parallelization of search strategy that makes it computationally cost-effective compared to general parallel pattern search (PPS) algorithms.

To be able to understand APPS and its associated advantages, one needs to first get familiar with PPS. This section further presents the outline of both PPS and APPS algorithms. Basic strategy for PPS is as following (Kolda and Torczon, 2003):

[5]This section is adapted and modified from author's specialization project (Chaturvedi, 2020).

Initialization:

- Set the iteration k=0
- select a step-length control parameter Δ_o
- select a stopping tolerance tol
- select a set of p search directions $D = d_1, ..., d_p$
- select a starting point x_0 and evaluate f(x_0)

Iteration:

- compute $x_k + \Delta_k d_i$ and evaluate f($x_k + \Delta_k d_i$), for $i = 1, ..., p$, concurrently.
- Determine x_+ and f(x_+) such that

$$f(x_+) = \min f(x_k + \Delta_k d_i) \quad \text{where} \quad i = 1, ..., p$$

- if $f(x_+) < f(x_k)$ then update

$$(x_{k+1}) = \begin{cases} \mathrm{x}_k + \Delta_k d_i\ , & \text{if} \quad k \in S \\ \mathrm{x}_k\ , & \text{otherwise} \end{cases}$$

$$\Delta_{(k+1)} = \begin{cases} \Delta_k, & \text{if} \quad k \in S \\ \theta \Delta_k, & \text{otherwise} \end{cases}$$

where S is set of successful iterations θ is contraction factor lies between 0 and 1, and λ is expansion factor, we have an iterate $x_k \in \mathbb{R}^n$ and a step length control parameter $\Delta_k > 0$. It is assumed that at the end of iteration k, all the processors know the best point x^k, where $f(x^k)$ is the best known value of function f.

For PPS, each of p processors oversee a single search direction in the set D. In PPS, the only communication among processors is the reduction of next step, where all the processors participating in compaction contribute their value for objective function and optimization variable. In short, reduction operation is synchronization point for PPS as it returns minimum value of objective function and corresponding value of optimization variable. In contrast to PPS that relies on a global reduction operation to synchronize all critical values, APPS relies on non blocking broadcasts to exchange information between processors and in APPS, every process decides what to do next based only on its current *local* information (Hough et al., 2001).

Asynchrony is useful as number of available processors may not be an integer of batch size, and/or execution time for function evaluation may not be the same for all the processors (Baumann, 2015).

2.3 Smart Proxy Model

Proxy models are "cheaper" and efficient alongside or alternative to commercial reservoir simulators. Proxy models can be developed based on statistical methods or based on artificial intelligence. SPMs are the proxy models based on artificial intelligence. The terms "surrogate model", "response surface model", and "meta-model" are also used alternatively for smart proxy model. The author uses proxy model and SPM terms interchangeably in this study.

Figure 2.5 shows a general workflow to construct SPMs. SPM is objective specific; therefore, it is important to identify the objective and accordingly design and develop SPM that can achieve that goal. In this project, the author develops SPMs capable of predicting field oil production rate (FOPR), field water production rate (FWPR) and field water injection rate (FWIR) at any given time. These SPMs are to be used for well control optimization study. Chapter 3 provides detailed description of steps for constructing SPM.

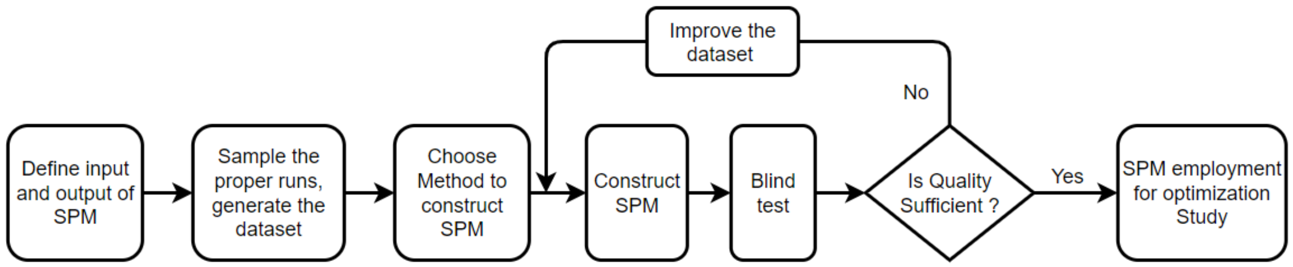

Improve the dataset
No
Define input and output of SPM
Sample the proper runs, generate the dataset
Choose Method to construct SPM
Construct SPM
Blind test
Is Quality Sufficient ?
Yes
SPM employment for optimization Study

Figure 2.5: General Workflow of developing SPM for Optimization Study.

2.3.1 Methods of Developing Proxy Models

Proxy models are used in many fields of science to approximate numerical models. In reservoir engineering, it is most widely used in reservoir simulation for probabilistic forecasting, risk analysis, sensitivity analysis, assisted history match, and production optimization (Jaber et al., 2019). The two main methods for constructing proxy models are discussed below:

Based on Statistical Methods

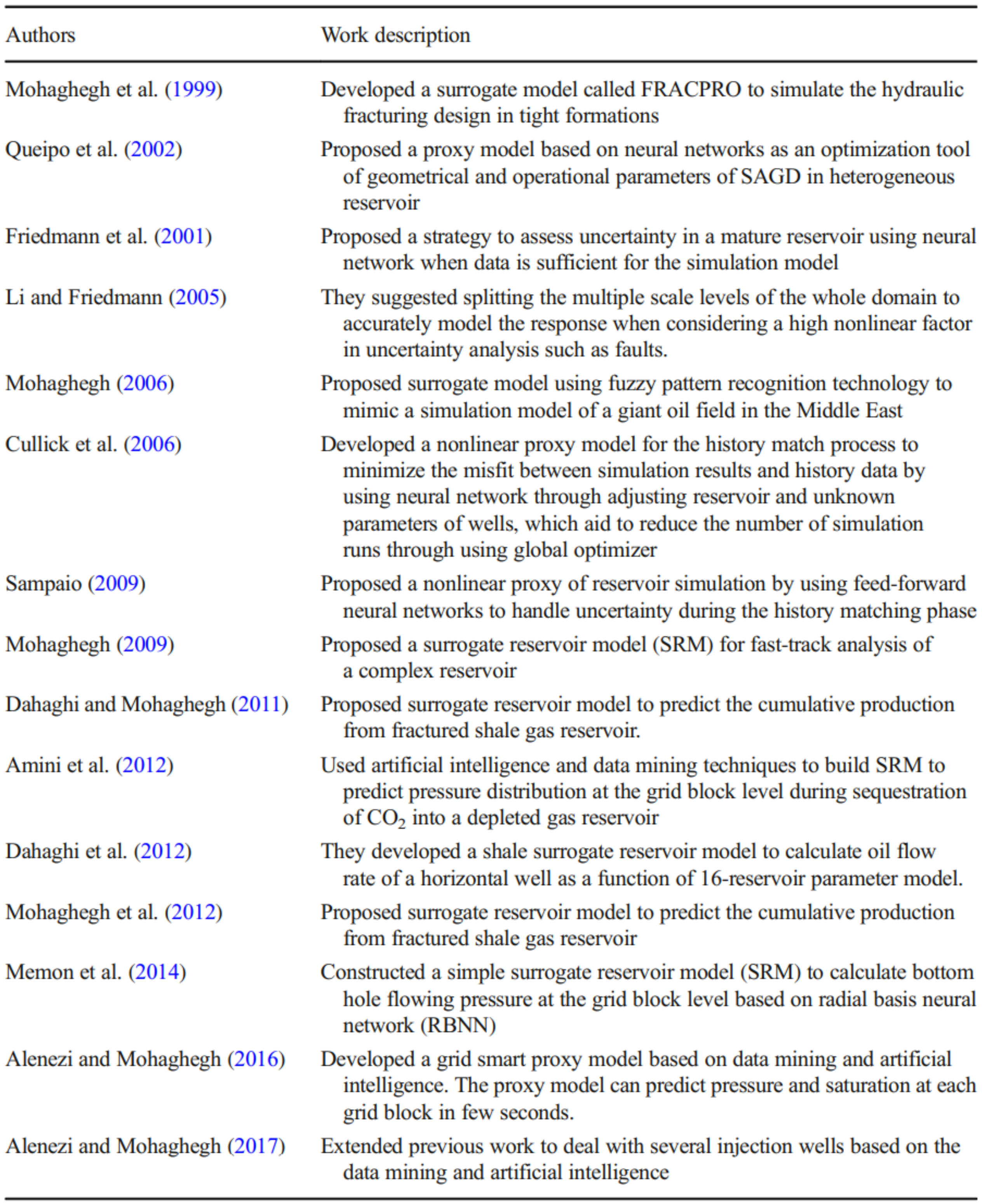
Numerous studies developed statistical proxy models as the approximations of the existing simulation models. The foundation of this approach was achieved by utilizing the design of experiments (DOEs) with response surface methodologies (RSMs) (Jaber et al., 2019). Denney (2010) applied polynomial regression model, multivariate kriging model and thin-plate splines model for history matching, production optimization and probabilistic forecast of oil recovery, and also commented on performances of these models for the tasks mentioned above. **Table 2** in (Jaber et al., 2019) summarizes history of application of proxy models based on statistical methods from various studies conducted from the year 1990 to 2018.

Based on Artificial Intelligence

Artificial intelligence-based modeling approaches in reservoir engineering aim to find the complex relationship between input-output parameters involved in fluid flow in porous media. These approaches have the flexibility to be used in various problems in petroleum and reservoir engineering disciplines such as field development planning, uncertainty analysis, optimization study, history matching, to mention a few. Although, for some cases, there might exist mathematical representation of physical phenomena to be considered for analysis purposes, the required computational efforts to carry out such analysis may make these mathematical formulations impractical (Amini, 2015).

Surrogate Reservoir Model (SRM) was introduced and developed in 2006 (Mohaghegh et al., 2006). One of the major advantages of SRMs, compared to conventional geostatistical techniques, is the small number of simulations required for their development. SRM can be developed by a certain set of realizations and be validated by another set of independent realizations (Mohaghegh et al., 2006). One key issue in the development of SRM is to realize that it is impractical to develop a global SRM that is capable of replicating all the functionalities of a reservoir simulation model (Mohaghegh et al., 2009). Figure 2.6 presents the history of proxy models based on artificial intelligence in the petroleum industry, developed by many researchers. Such models have been developed for the prediction of reservoir performance, assisted history matching, and optimization workflow.

This study uses ANN for constructing SPMs. Therefore, next section covers the basic principles of Artificial Intelligence and practical aspects of building Artificial Neural Network from scratch.



Authors	Work description
Mohaghegh et al. (1999)	Developed a surrogate model called FRACPRO to simulate the hydraulic fracturing design in tight formations
Queipo et al. (2002)	Proposed a proxy model based on neural networks as an optimization tool of geometrical and operational parameters of SAGD in heterogeneous reservoir
Friedmann et al. (2001)	Proposed a strategy to assess uncertainty in a mature reservoir using neural network when data is sufficient for the simulation model
Li and Friedmann (2005)	They suggested splitting the multiple scale levels of the whole domain to accurately model the response when considering a high nonlinear factor in uncertainty analysis such as faults.
Mohaghegh (2006)	Proposed surrogate model using fuzzy pattern recognition technology to mimic a simulation model of a giant oil field in the Middle East
Cullick et al. (2006)	Developed a nonlinear proxy model for the history match process to minimize the misfit between simulation results and history data by using neural network through adjusting reservoir and unknown parameters of wells, which aid to reduce the number of simulation runs through using global optimizer
Sampaio (2009)	Proposed a nonlinear proxy of reservoir simulation by using feed-forward neural networks to handle uncertainty during the history matching phase
Mohaghegh (2009)	Proposed a surrogate reservoir model (SRM) for fast-track analysis of a complex reservoir
Dahaghi and Mohaghegh (2011)	Proposed surrogate reservoir model to predict the cumulative production from fractured shale gas reservoir.
Amini et al. (2012)	Used artificial intelligence and data mining techniques to build SRM to predict pressure distribution at the grid block level during sequestration of CO_2 into a depleted gas reservoir
Dahaghi et al. (2012)	They developed a shale surrogate reservoir model to calculate oil flow rate of a horizontal well as a function of 16-reservoir parameter model.
Mohaghegh et al. (2012)	Proposed surrogate reservoir model to predict the cumulative production from fractured shale gas reservoir
Memon et al. (2014)	Constructed a simple surrogate reservoir model (SRM) to calculate bottom hole flowing pressure at the grid block level based on radial basis neural network (RBNN)
Alenezi and Mohaghegh (2016)	Developed a grid smart proxy model based on data mining and artificial intelligence. The proxy model can predict pressure and saturation at each grid block in few seconds.
Alenezi and Mohaghegh (2017)	Extended previous work to deal with several injection wells based on the data mining and artificial intelligence

Figure 2.6: History of proxy models based on artificial intelligence (Jaber et al., 2019).

2.4 Artificial Intelligence

AI, first introduced by McCarthy in 1955, is a universal field which encompasses a vast variety of subfields (Russell and Norvig, 2003). AI is a broad topic that consists of different fields, from machine learning to expert systems with a combination of computer science, physiology, and philosophy (Mohaghegh, 2017). AI also enables machines to adjust their "knowledge" based on new information that was not used to train these machines.

There are several definitions to explain AI; one may define AI as the ability that can be conferred to computers which enable these machines to understand data, learn from data, and make decisions based on the patterns hidden in the data that could be otherwise difficult or almost impossible for humans to identify manually. Artificial intelligence and machine learning (ML) are often used interchangeably, which is not correct.

Machine learning is a subfield of AI that focuses on developing computer systems that can access data and use it to learn themselves. ML can be defined as the use and development of computer systems that can learn and adapt without following explicit instructions using algorithms and statistical models to analyze and draw inferences from patterns in data.

Machine learning algorithms can be divided into supervised and unsupervised based on learning methods, that are explained below.

Supervised Learning

Supervised learning is the learning in which human teaches machines with well-labeled data (or data that is tagged with the correct answer). After the training process, machine produces correct outcome from the labeled data. Supervised learning algorithms are divided into two parts based upon their outputs:

1. Classification: It means to group the output inside a class, or when output is a categorical variable. If the algorithm tries to label input into two distinct classes, it is called binary classification. Selecting between more than two classes is referred to as multi-class classification. Classification is used in voice recognition and image classification. Few popular classification algorithms are Logistic Regression, Neural Network, and Decision tree.

2. Regression: It predicts a single output that is a real value from the trained data. Regression technique is used in prediction of quantities, size, and value, to mention a few. Most common regression algorithms are linear, support vector and Poisson.

Unsupervised learning

As the name suggests, unsupervised learning is when no supervisor is provided; instead, the machine learns on its own to discover the information hidden in the data. It mainly deals with unlabeled data. Followings are two categories of unsupervised learning algorithms:

1. Clustering: It deals with finding a structure, pattern, or groupings in a collection of uncategorized or unlabeled data.
2. Association: Association rules allow one to discover and establish associations amongst data objects inside large databases. This deals with discovering relationships between variables in the databases, such as people who buy a new house most likely buy new furniture.

2.4.1 Artificial Neural Network (ANN)

An artificial neural network is an information processing system inspired by the biological nervous system, such as how brain processes information. It is composed of highly interconnected processing units (also called neurons) working together to solve a specific problem. As presented in Fig. 2.7, biological neurons or simply neurons are the fundamental units of the brain and nervous system.

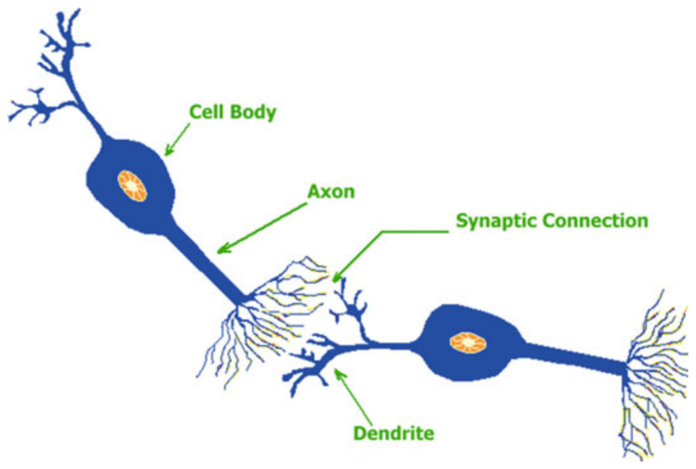


Figure 2.7: Schematic diagram of two bipolar neurons (Mohaghegh, 2017).

To better understand working procedure of the neural network, be it biological or artificial, it is vital to discuss units that make it up.

Cell body: A typical neuron contains a cell body where nucleus is located. It carries out bio-chemical transformation necessary to the life of neurons. It processes incoming signals over time and provides output to be sent out further.

Dendrites: Dendrites are fine, hair-like tubular structures branching out into a tree around the cell body. Input signals enter the cell body through dendrites, whereas dendrites receive input signals from other neurons' synapses.

Axon: It is a long, thin, tubular structure that transmits outgoing signals from the cell body.

Synapse: Neurons are connected in a complex structure. The point between two neurons, where termination of axon of one neuron comes into proximity of dendrites of another neuron, is called synapse (Mohaghegh, 2017).

Figure 2.8 represents a single neuron with inputs, connections, bias and output. For one single observation, x_0, x_1, x_2,...,x_n represents n independent input variables. Each of these input variables are multiplied by their respective weights given by w_0, w_1, w_2,...,w_n. Weights are almost always learned from the data. The inputs are summed, and a bias value is added that allows the activation function to shift up or down. Neuron takes this weighted sum and puts it through a so-called activation function to generate output.

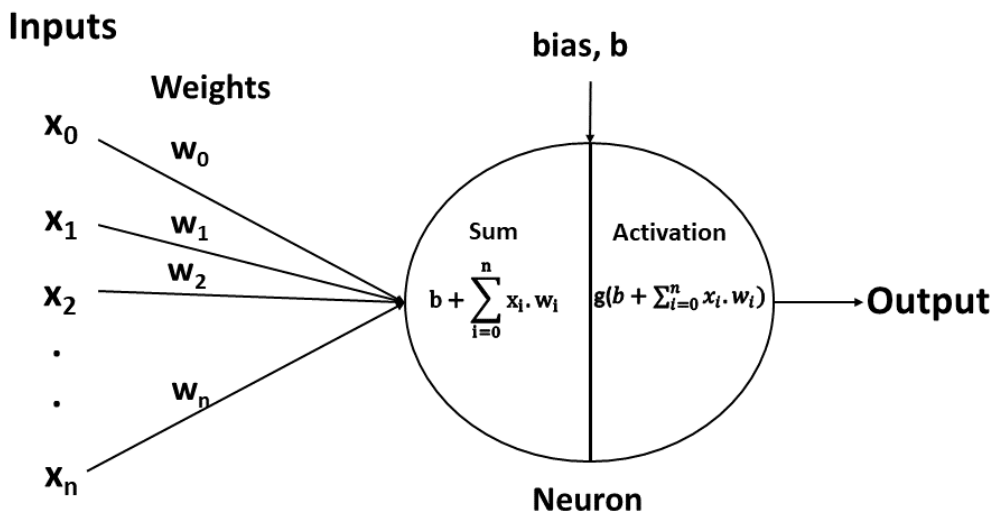


Figure 2.8: Schematic diagram of a single neuron which receives inputs with weights and bias, sums all inputs and utilises an activation function to calculate a scalar value which is sent as output from the neuron.

A complete neural network will have many layers and neurons. Figure 2.9 illustrates a three-layer neural network. A input layer does not count as a layer because it only receives inputs fed to neurons present in hidden layers. Neurons are illustrated by circles, input layer by inputs

x, hidden layers by neurons in the two middle columns and output layer is final neuron on the right. Neurons in the first hidden layer receive inputs and calculate output based on each input variable's weights, bias, and activation function. This output is input for neurons in the second layer. Hence, for this ANN, the first layer of neurons is making three simple decisions by weighing the input variable. While each of the neurons present in the second layer makes a decision by weighing up the results from the first layer of decision-making. This allows neurons in the second layer to make a decision at a more complex and abstract level than neurons in the first layer (Nielsen, 2019).

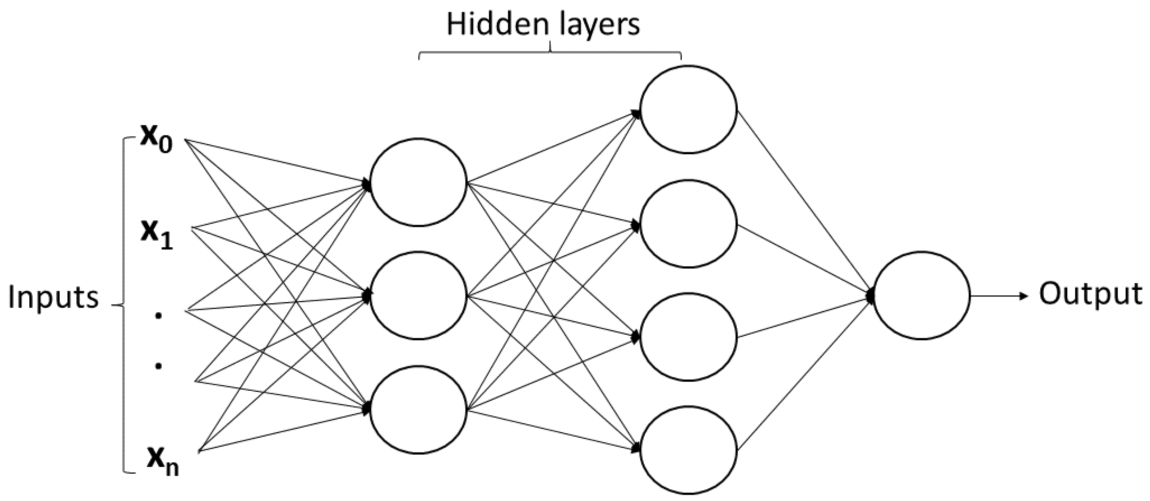


Figure 2.9: Schematic diagram of a Neural Network with two hidden layers, input layer and output layer. The fully-connected Neural Network shows how all neurons in each layer are connected to both the neurons in the previous layer and the next layer.

Activation Function

An activation function is vital for an ANN model to learn and convert an input signal of a neuron to an output signal. Once neuron calculates the weighted sum, an activation function (g as shown in Fig. 2.8) is used to decide if the neuron is activated or not. If neuron is activated, then it fires output to next neurons; otherwise, it does not. There are several options when choosing an activation function. However, it is wise to choose a non-linear activation function as it can make even a two-layer neural network approximate any given function provided sufficient number of neurons in the hidden layer (Ketkar, 2017a). In linear activation function, input is multiplied by 1, therefore it ranges in $[-\infty, \infty]$. This is since the linear activation function does not alter the weighted sum of the input and instead returns the result straight. Therefore, in case of a linear function, hidden layers lose their advantage, and in turn, network performs like a single-layer network. Figure 2.10 displays three commonly used non-linear activation functions.

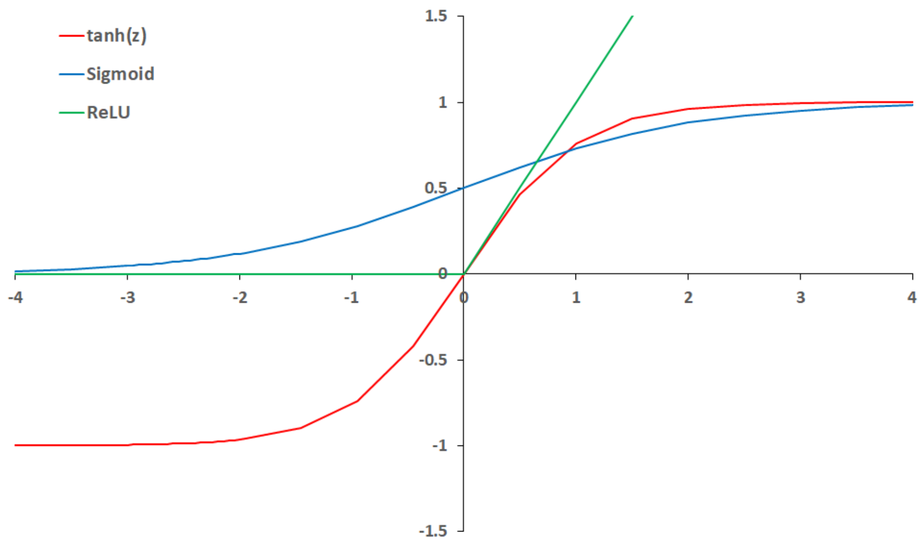


Figure 2.10: Representation of three commonly utilized activation functions in neural network: Tangent Hyperbolic, Sigmoid, and ReLU.

Sigmoid and Hyperbolic tangent functions are continuous, have a finite range and are symmetric around the origin. These functions transform the input (z) as defined by Eqs. 2.5 and 2.6, respectively. The main difference between the two functions is the range, with sigmoid output ranging in [0,1], and hyperbolic tangent output ranging in [-1,1]. Both functions activate neurons almost always except for the 0 input value. Thus, a large neural network with many layers and neurons will lead to a high computation time. To overcome this problem, a rectified linear unit (ReLU) function can be utilized which is defined by Eq. 2.7; zero output for all the negative input implying that it won't activate neurons if the input is negative value (Nielsen, 2019). ReLU is also non-linear with output ranging in $[0,\infty]$.

$$f(z) = \frac{1}{1+e^{-z}} \tag{2.5}$$

$$f(z) = tanh(z) \tag{2.6}$$

$$f(z) = max(0,1) \tag{2.7}$$

Loss function

Loss function computes disagreement between expected output and predicted output (Ketkar, 2017a), it is also known as cost function. It is used to optimize the weights, and the process of optimization is also called loss minimization. There are several ways to find loss function; one is mean squared error (MSE). MSE is given by Eq. 2.8 and is recommend to use for regression problems (Ketkar, 2017a).

$$J(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y_i})^2 \tag{2.8}$$

Here, J is the loss function, n is the number of trained data, y is a vector of actual output, and $\hat{y}$ is a vector of predicted output. It is worth noting that cost function is a single value, not a vector

This study also uses MSE as loss function. Optimization algorithms use loss function to find optimum weights in the neural network corresponding to minimum loss function. In this study, the author uses the Adam optimizer (explained further down).

Adam Optimizer

Adam, introduced by (Kingma and Ba, 2017), is an optimization algorithm that can be used to update network weights iterative based in training data. Adam is derived from Adaptive Moment Estimation and is not an acronym. "Adam" is not the same as traditional stochastic gradient descent. For all weight updates, stochastic gradient descent maintains a single *learning rate* (called alpha), which does not fluctuate during training. Kingma and Ba (2017) described Adam as combining the advantages of two other extensions of stochastic gradient descent, Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp). Adam is a widely used optimizer for neural network nowadays.

2.4.2 Practical Aspects of Artificial Neural Network

This section covers practical aspects of building neural networks and common problems faced in the process. Several Python libraries are utilized to build the Neural Network, such as *Keras*, *Scikit-Learn*, *Numpy* and *Pandas*. In the following sections, the key concepts used for code implementations are explained.

Data Splitting

Before feeding data to neural network, dataset is usually split into following categories defined below:

1. **Training Dataset**: The dataset that is used to train the model. Model *sees* and *learns* from this data. The desired output in the training set is used to adjust weights between its neurons in the networks (Mohaghegh, 2017).

2. **Validation Dataset**: To avoid overtraining or memorization of the neural network, it is a common practice to stop the training process and apply the network to the validation dataset. The validation dataset is, in turn, used to fine-tune the hyperparameters. Values of hyperparameters are set before the learning process begins, and optimum hyperparameters are found by minimizing error on the validation dataset. Thus, model occasionally *sees* this data, but never *learns* from it. Output of this dataset is not presented to the network during training; therefore, one can comment on network's performance by predicting output for validation dataset (Mohaghegh, 2017).[6].

3. **Test Dataset**: This dataset plays no role during training or validation. It is set aside from the beginning and only used once a model is completely trained. However, these datasets help to validate the robustness of the predictive capabilities of the neural network (Mohaghegh, 2017). Hence, model never *sees* and *learns* from this data in the training process.

4. **Blind Test**: Similar to the test dataset, this dataset plays no role in the training or validation process. It is used to evaluate the performance of a model. In contrast to the previous three datasets generated together and then split into three, it is generated separately as a final performance check of prediction models.

Data Scaling or Feature Scaling

Inputs to the neural network are also known as features. A dataset can consist of features highly varying in magnitudes, units, and range. Data scaling involves transforming features in a standard range to avoid dominance of features with larger numeric values over features with smaller numeric values (Singh and Singh, 2020). Ng (2021) commented that highly varying features in scaling slow down the optimization of the prediction. Therefore it is an essential data preposessing step. Two common methods to scale the data are explained below:

- Min-Max Normalization (MMN): MMN is a common way of normalization. Features can be *normalized* by rescaling values into a range of [0,1], or [-1,1]-if feature contains negative

[6](Mohaghegh, 2017) discussed that database should be divided into training, calibration, and validation. In this study, to avoid confusion, the calibration set is termed as the validations sets, whereas validation set is referred to as testing set.

values. This method rescales raw data to a predefined upper and lower bound linearly. The formula of MMN is given by Eq. 2.9. This study uses MMN method to scale the data.

$$x_i' = \frac{x_i - min(x_i)}{max(x_i) - min(x_i)} \tag{2.9}$$

where *min* and *max* are minimum and maximum values of i^{th} feature, respectively.

- Standardization: Standardization is another common way of rescaling features such that the distribution is centered around 0 with standard deviation of 1. This technique uses statistical mean and standard deviation of raw data for data scaling. The formula is expressed in Eq. 2.10.

$$x_i' = \frac{x_i - \mu}{\sigma} \tag{2.10}$$

where μ and σ denote mean and standard deviation of i^{th} feature, respectively.

Underfitting and Overfitting

Underfitting, also called high bias, is caused by a model that fits poorly with the data trend. This problem usually arises from a function that is too simple. On the contrary, overfitting, also known as high variance, is when the model fits the training data but fails to generalize well to predict new data. Overfitting problem arises with too many features, and the model tries too hard to fit the training set that results in a complicated function. Overfitting issues can be handled in many ways; some approaches are listed below:

- Reducing number of features: ANN becomes complicated with the increase in the number of features. This, in turn, causes the problem of overfitting. One possible way to overcome this problem is by selecting only the valuable features. The choice should be made carefully, as discarded features might contain some helpful information.

- Early stopping: One can evaluate the performance of the model at each iteration in training process. Models continue to improve up to a certain number of iterations. Early stopping strategy is used to avoid phenomenon "learning speed slow-down", which means that accuracy of algorithms stops improving after some point.

- Regularization: Finding valuable features is not always easy, so one can limit them all by minimizing cost function of the neural network. Regularization can be realized as a modification to the model that aims to improve the error on validation set by systematically limiting complexity of the model (Ketkar, 2017b).

Hyperparameter Optimization

Hyperparameter is a parameter whose value is determined before the learning process starts, such as the learning rate, and which cannot be learned by training. Other parameters (typically node weights) are, on the other hand, learned by training. The problem of selecting a set of suitable hyperparameters for a learning algorithm is known as hyperparameter optimization (HPO) or hyperparameter tuning in machine learning.

Some examples of possible hyperparameters are given below (Ying, 2019)

- Activation function: defines how a neuron or a group of neurons activates based on inputs and biases, recall section 2.4.1.
- Neuron counts: the number of neurons in a layer.
- Number of epochs: The number of times all the training samples have been passed through the network during the training process.
- Number of layers: also called hidden layers.
- Mini-batch size: number of training dataset in each gradient descent.
- Learning rate: step-length for gradient descent update.
- Learning rate decay: incrementally decaying learning rate parameter throughout the training to prevent overfitting.
- Dropout (Srivastava et al., 2014): dropping out some input connections to some neurons by a probability to make the ANN represent features more evenly.
- Early stopping: stopping ANN training when training and validation error starts to diverge from each other to prevent over-fitting.

Once user selected which hyperparameters to include to optimize performance of ANN or refine ANN model, user defines range of each parameter to create a solution space. Then, based on the optimization function and optimization algorithm, hyperparameters are fine tuned. There are several approaches for HPO, some are described below:

1. **Grid Search**: Grid search, which is an exhaustive search over a manually chosen subset of a learning algorithm's hyperparameter space, has been the conventional method of accomplishing HPO. The loss is recorded after testing every conceivable combination of parameters in the defined subset of hyperparameters. The hyperparameter combination that produces the smallest loss function is then chosen. The need to evaluate every possible configuration slows down the entire grid search process. As a result, a subset of configurations can be chosen manually by intuition before or during the grid search process. This, however, introduces reproducibility issues and requires the HPO to be performed sequentially.

2. **Bayesian Optimization**: *Bayesian Optimization* is a systematic methodology based on the Bayes Theorem for directing an efficient and successful search of a global optimization problem. It works by first creating a probabilistic model of the objective function, known as the "surrogate function", which is then effectively searched with an acquisition function before selecting suitable samples for assessing the real objective function. Surrogate function is sometimes referred to as a response surface because it is a high-dimensional mapping of hyperparameters to the probability of a score on the objective function. This study uses Bayesian optimization by importing Gaussian-based function **gp_minimize** from *Skopt* library (Louppe et al., 2016).

2.4.3 Model Fitting and Evaluation

Assessing the reliability of the proxy model is regarded as one of the most important steps toward developing a reliable model. Inaccurate proxy models can result in poor quality optimization analysis. Metric functions are used to evaluate the efficiency of the model. It is identical to the loss function, except the results from metrics evaluations are not used in training the model (Keras, 2021). In addition to trained ANN validation during HPO, performance measures are often used to evaluate accuracy of the trained models and application of these models. Most researchers, for the sake of simplicity, present only one performance measure to indicate the "goodness" of a particular network's performance. However, there is no agreement on which measure should be reported and comparing techniques and results from different researchers is nearly impossible. Some popular performance metrics are descried below with the formulas required for calculation.

R^2 Score

The coefficient of determination, R^2, is used to examine how differences in one variable can be explained by differences in another. The R^2 score varies between 0 and 1. It is given by the formula in Eq. 2.11.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \bar{y})^2}{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{2.11}$$

Here, n is the number of trained data, y is a vector of actual output, and $\hat{y}$ is a vector of predicted output, and $\bar{y}_i$ is mean of actual data.

So, a R^2 score of 1 implies that y and $\hat{y}$ are perfectly correlated, i.e., there is no variance. Low R^2 score indicates a low level of correlation, suggesting that the regression model is invalid, but not in all cases.

MAPE

Mean absolute percentage error (MAPE) is calculated by Eq. 2.12. It measures how accurate a prediction model is. It measures in percentage; high numbers are bad, and low numbers are good. As shown in the formula, MAPE divides each error by expected value individually, so it is skewed: high errors for low-expected values significantly impact MAPE. As a result, optimizing MAPE will produce an odd forecast that will almost certainly undershoot expected value. Therefore, MAPE is not used for hyper-parameter optimization.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{y_i} \times 100 \tag{2.12}$$

NRMSE

The Root Mean Squared Error (RMSE) is a common way of measuring the quality of the model's fitting in statistical modeling, particularly regression analyses. By squaring the errors when calculating the RMSE, a larger penalty is applied to large errors (clearly incorrect responses), even if they are few (Twomey and Smith, 1995). Normalizing the RMSE facilitates comparing datasets or models with different scales. Therefore, NRMSE (Normalized Root Mean Squared Error) is used to evaluate SPM performance. NRMSE is calculated using Eq. 2.13 by dividing RMSE by $\bar{y}$.

$$NRMSE = \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}}{\bar{y}} \times 100\% \tag{2.13}$$

In percentage terms, this metric shows how close the model prediction is to the actual value. The model's performance improves as the NRMSE decreases. Each error influences NRMSE that is proportional to the square of the error; thus, larger errors have a disproportionately large effect on NRMSE. As a result, NRMSE is sensitive to outliers, implying that it should be more useful when large errors are especially undesirable. This metric is well-suited for this study because large errors can result in NPV differences worth millions of dollars.

MAE

MAE calculates the arithmetic average of absolute errors as shown in Eq. 2.14, also considers large errors but does not weight them more heavily unlike RMSE. This metric is considered appropriate for this study as errors on each data point are distributed evenly, hence, MAE can be considered a good indicator of prediction errors.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \times 100 \tag{2.14}$$

Chapter 3

Methodology

This chapter presents how the different technologies described in the previous chapter are used to perform well control optimization using methods from the AI field.

3.1 Workflow

Figure 3.1 shows the main workflow of this master's thesis.

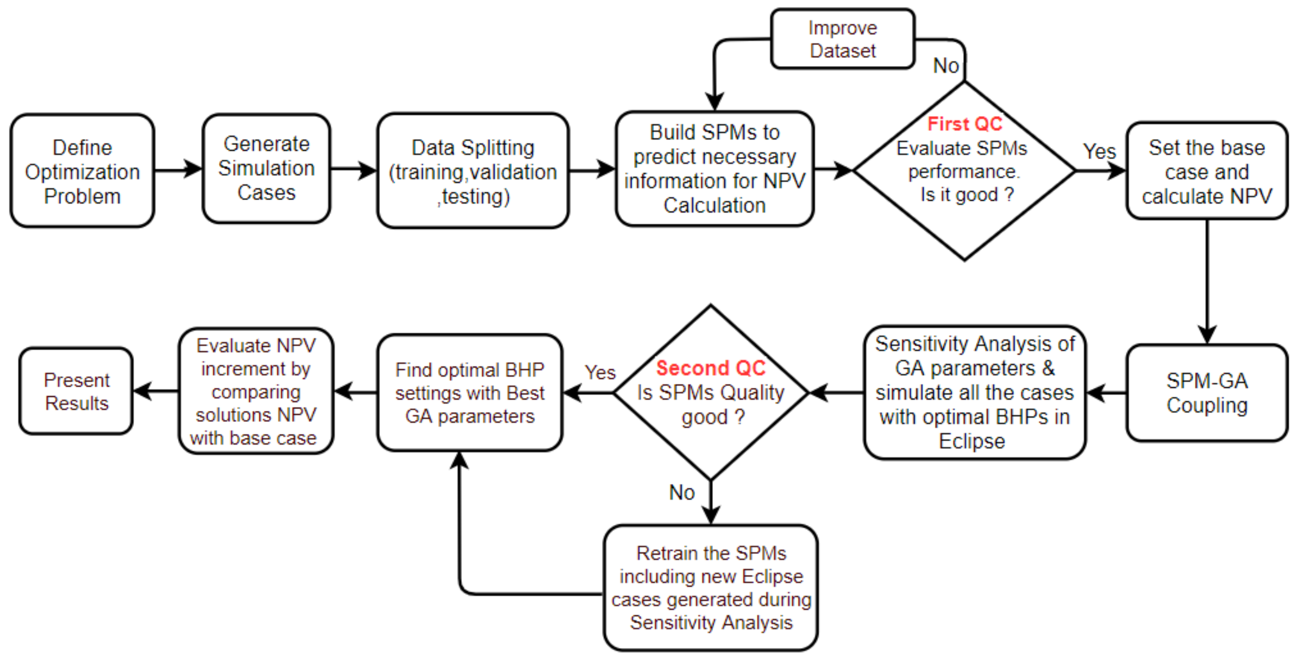

Define Optimization Problem
Generate Simulation Cases
Data Splitting (training,validation ,testing)
Build SPMs to predict necessary information for NPV Calculation
Improve Dataset
No
First QC Evaluate SPMs performance. Is it good ?
Yes
Set the base case and calculate NPV
SPM-GA Coupling
Sensitivity Analysis of GA parameters & simulate all the cases with optimal BHPs in Eclipse
Second QC Is SPMs Quality good ?
No
Retrain the SPMs including new Eclipse cases generated during Sensitivity Analysis
Yes
Find optimal BHP settings with Best GA parameters
Evaluate NPV increment by comparing solutions NPV with base case
Present Results

Figure 3.1: The main workflow of master's thesis.

The first step is to define the optimization problem. Following that, necessary information is extracted from the simulation runs on reservoir model to create a dataset. This dataset is then partitioned into three subsets as discussed in section 2.4.2: training, validation, and test. The following step is to develop an ANN model capable of forecasting oil production rates, water production rates, and water injection rates based on sets of well control values (BHPs). After that, the predictive capability of each proxy model is evaluated using performance metrics. If the quality is insufficient, additional training data is fed into the neural network. After determining that the proxy model's performance is acceptable, a base case is defined, and the proxy model's NPV is calculated. Next, the SPMs and GA are coupled to determine the optimal well control values. SPM-GA coupling is then used for sensitivity analysis of GA configuration parameters. After that, the optimal BHP configuration cases from each sensitivity run are simulated using the reservoir simulator, and NPV is calculated based on simulator outputs. Subsequently, a second quality check on proxies is done by comparing NPV from simulator's predictions with NPV from proxy models' predictions. If quality is good, then optimization is run again with GA best configuration settings and the optimal BHPs are found. Otherwise, proxies are retrained by including new information from sensitivity runs on simulator into training dataset. Finally, the NPV from the optimizer result is compared to the NPV from the base case, and the result is analyzed and discussed.

Following section presents and discusses the details of the reservoir model used for this study.

3.2 Reservoir Model [1]

Based on the requirements from PhD candidate, Olympus model is selected to implement the workflow described in previous section and achieve the objective of this study. Olympus is a synthetic reservoir model, inspired by virgin oilfield in the North Sea, has been developed for the purpose of a benchmark study on field development optimization under geological uncertainty (Fonseca et al., 2017). The reservoir is 50 m thick and divided into 16 layers. The field is 9 km by 3 km and is bounded on one side by a fault. The reservoir contains two zones that are separated by an impermeable shale layer. The top reservoir zone contains fluvial channel sands embedded in floodplain shales, and the bottom reservoir zone consists of alternating layers of coarse, medium and fine sands that are inclined with respect to the general structural dip of the field, so-called clinoforms (Fonseca et al., 2018). There are in total 50 model realizations to incorporate geological uncertainty. Model consists of grid cells of approximately 50m × 50m × 3m each. The model has 341,728 cells out of which 192,750 are active. Reservoir consists of four

[1] This section is adapted and modified from author's specialization project (Chaturvedi, 2020).

Table 3.1: Facies types and range of uncertain properties distribution for the OLYMPUS reservoir models (Fonseca et al., 2018).

Facies Type	Zone Present	Net-To-Gross	Porosity Range	Permeability Range [mD]
Channel Sand	Upper	0.8-1	0.2-0.35	400-1000
Shale	Upper and Barrier	0	0.03	1
Coarse Sand	Lower	0.7-0.9	0.2-0.3	150-400
Medium Sand	Lower	0.75-0.95	0.1-0.2	75-150
Fine Sand	Lower	0.9-1	0.05-0.1	10-50

different facies that have been modeled in different layers as shown in Table 3.1: channel sand, coarse sand, medium sand and fine sand.

The reference strategy consists of 10 producers and 6 injectors also shown in Fig. 3.2 that operates on a pressure constraint. The placement of wells in reference strategy is result of a manual trial and error exercise based on engineering judgement for a chosen realization, thus well placement is probably not optimal over all the realizations (Fonseca et al., 2017). The author works with Olympus_49 realization in this study. However, the author has built proxy models for two more realizations as required for the PhD research project. Figure 3.3 represents visualization of Olympus_49 model.

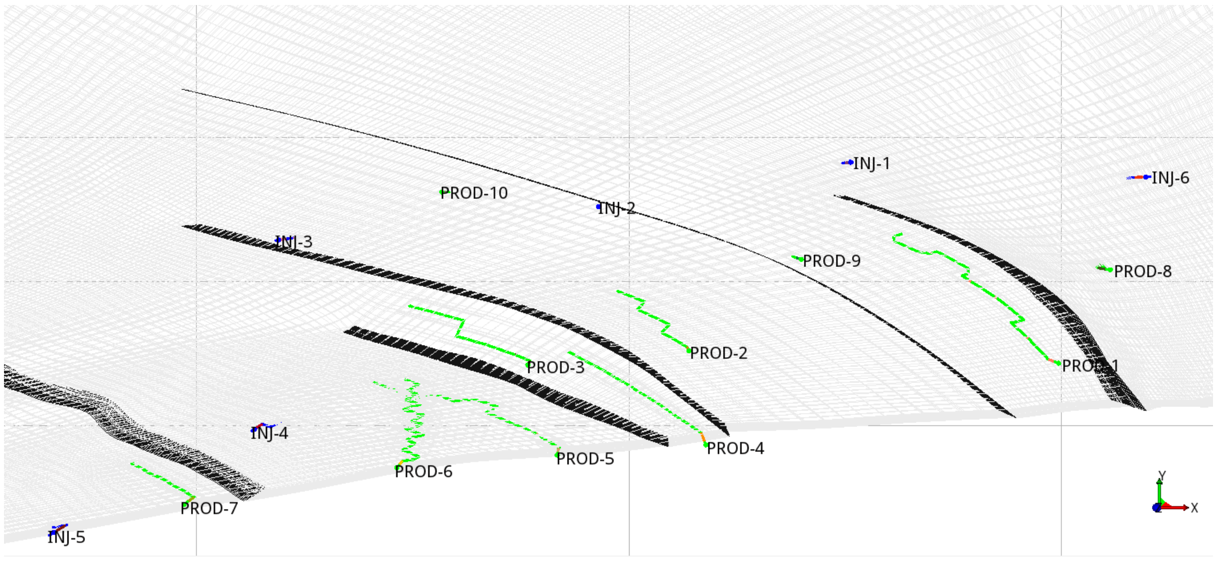
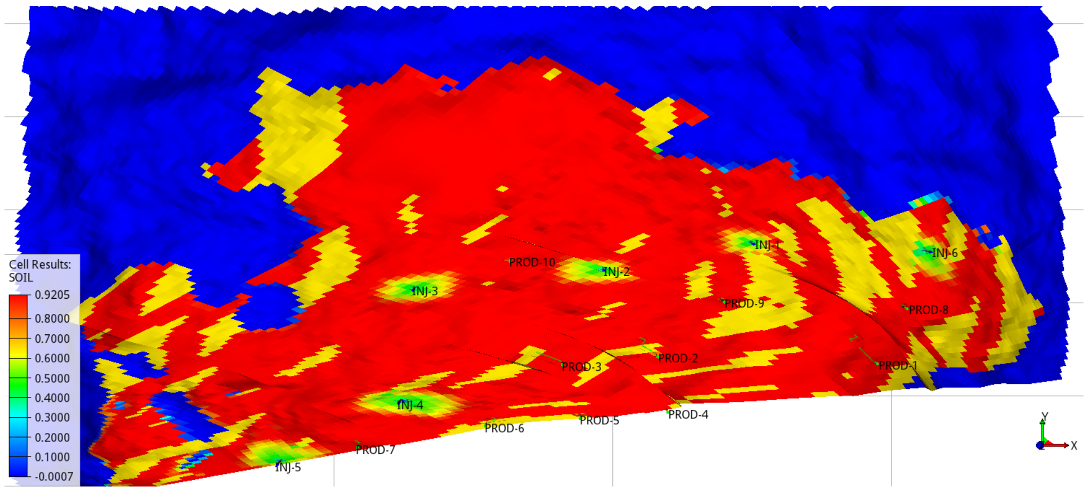


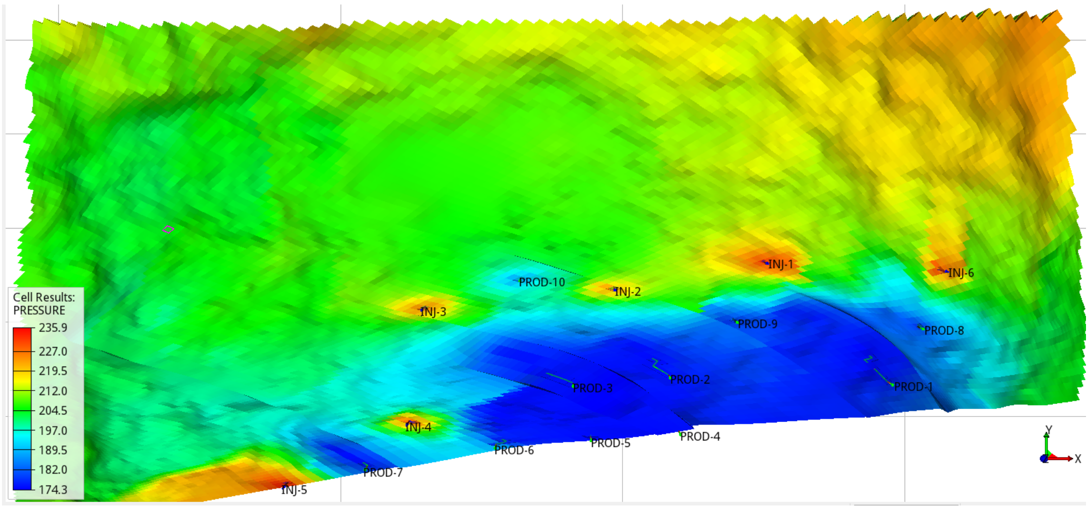
Figure 3.2: Well configuration for Olympus_49 realization.

Depth of OWC was determined to be at 2090m, with an in-situ hydrostatic pressure of 206 bara from the available exploration well logs (Fonseca et al., 2017).

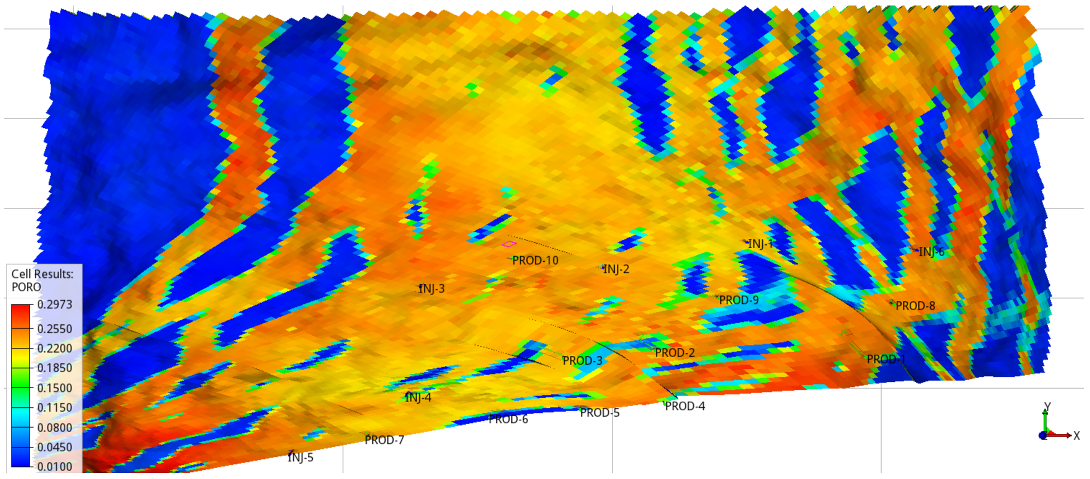
Table 3.2 shows operational constraints for wells in Olympus model defined by developers of Olympus challenge. This project deals with BHP constraint that is discussed in detail in section



(a) Initial oil saturation distribution.



(b) Initial pressure distribution.



(c) Porosity distribution.

Figure 3.3: Visualizations of the Olympus_49 model at start time.

3.3.

Table 3.2: Operation constraints for wells in Olympus models.

Type	**Value**
Maximum platform liquid production rate (Sm^3/day)	14000
Maximum well oil production rate (Sm^3/day)	900
Maximum well water injection rate (Sm^3/day)	1600
Maximum Injector BHP (bar)	235
Minimum Producer BHP (bar)	150
Maximum dogleg severity ($^\circ/m$)	10/30
End of the production life (years)	20

3.3 Optimization Problem

The primary goal of a field development optimization problem is often to find a set of parameters represented by a vector of variable values, x, that yields the optimal (here maximum) objective function value. The number of control variables ($x_i, i = 1, 2, ...N_w$) together with number of wells defines number of dimensions of search space (N_x), $N_x = \sum_{i=1}^{N_w} x_i$. In this work, optimization variable is BHP for producers and injectors, and the objective function that we are optimizing is the net present value (NPV). Equation 2.1 discussed in section 2.2.1 is used to calculate NPV from simulator's outputs and proxy model's predictions. Table 3.3 presents some important parameters used in the NPV computation. The oil price chosen here is representative of current market. To solve well control optimization problem, the author divides 20 years of production period into twenty time intervals and controls the BHP for all producers and injectors for every year, so considering base case scenario with ten producers and six injectors, the total number of optimization variables are 320. The constraints for the optimization problem are presented in Table 3.4. BHPs range is chosen to avoid instances that lead to shut down of injector due to higher reservoir pressure than injector's BHP and shut down of producer due to lower reservoir pressure compared to producer's BHP.

Table 3.3: This table represents NPV Computation Parameters.

Type	**Value**
Oil price ($/bbl)	60
Cost produced water ($/bbl)	6
Cost injected water ($/bbl)	2
Annual discount factor	0.08
End of life cycle period (years)	20

Table 3.4: Optimization constraints (the pressure unit is bara).

	Base case	Min	Max
Injector BHP	235	200	280
Producer BHP	175	80	175

3.4 Building Smart Proxy Model

This section describes the procedure used to construct the SPMs in this study (refer to Fig. 2.5 for general workflow for proxy model development).

3.4.1 Defining input and output for SPM

Optimization problem discussed in section 3.3 forms the foundation for input and output selection of SPM. Therefore, BHPs and timesteps are fed as input into proxies, and outputs are variables required for NPV calculation (i.e., FOPR, FWPR, and FWIR). Timestep is considered as input to assist the proxy models in learning the dynamic flow behavior. Three proxies are constructed for each output variable. Figure 3.4 illustrates inputs given and output obtained from each proxy, and ANN architecture used for constructing proxies discussed already in section 2.4.1.

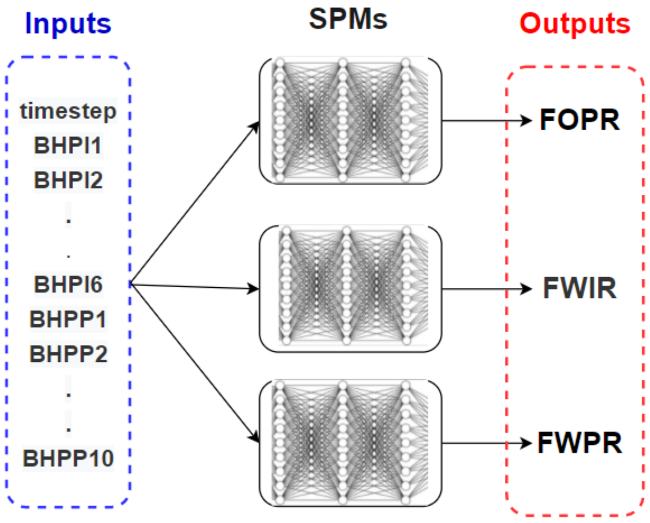


Figure 3.4: Representation of inputs and output of each SPM.

3.4.2 Generating the Dataset and Data Splitting

As proxy models need to learn from multiple scenarios to make a good prediction, lots of different scenarios are generated using Latin Hypercube Sampling (LHS) method and are simulated in Eclipse, a commercial reservoir simulator developed and maintained by Schlumberger Information Solutions (SIS).

In the reservoir simulator, BHP of producers and injectors are input by the user, and simulator calculates FOPR, FWPR and FWPR. As in one simulation case, 20 years of production life of reservoir results into 20 BHP values for each well, thus 320 different BHP values in total for 16 wells (10 producers and 6 injectors), also discussed in section 3.3. One timestep, Δt is defined for 1 year, thus one simulation case generates 20 data points for Olympus model. The limits of the BHPs are as stated in Table 3.4, i.e. within a range of [80,175] bara for producers and [200,280] bara for injectors. BHPs are sampled using Latin Hypercube Sampling (LHS) method to generate samples for training, validation and testing the proxy models.

Table 3.5 shows the data collected by simulating the base case (which is defined by 175 bara producer's BHP and 235 bara injector's BHP at all the timesteps). Unlike the base case, samples generated using LHS method consists of varying BHP across timesteps. Established Dataset is divided into training, validation and testing data, recall section 2.4.2 for definition of each data type.

Table 3.5: Information gathered from one simulation case (base case) run on Eclipse.

Timestep	**BHPs [bara]**								**Rates [Sm^3/day]**		
[year]	**I1**	**I2**	**...**	**I6**	**P1**	**P2**	**...**	**P10**	**FOPR**	**FWIR**	**FWPR**
1	235	235	...	235	175	175	..	175	1703	0.4024	1857
2	235	235	...	235	175	175	..	175	1714	70.83	1979
.	.	.	...	.	.	.	..	.	.	.	.
.	.	.	...	.	.	.	..	.	.	.	.
.	.	.	...	.	.	.	..	.	.	.	.
19	235	235	...	235	175	175	..	175	558	1748	2312
20	235	235	...	235	175	175	..	175	532	1748	2351

The proxy models are designed such that the correlation of various BHPs with FOPR, FWPR, and FWIR at various timesteps are expected to be learned during the training process. Recall from section 2.4.2, the training phase consists of three distinct processes: training or learning, validation, and testing. The best network architectures are achieved after a series of optimization processes involving network performance monitoring. The last step in developing the proxy models is testing using completely blind runs (that are not used during training). Therefore, five extra cases are designed for this purpose. This dataset confirms if the model can be generalized to other data or not.

The simulation results are stored in *.csv* files and later loaded into Jupyter Notebook as DataFrame using *Pandas* module. Study begins with 30 samples and the number of samples are increased subsequently until desired quality of proxy models are achieved. Following is a short description of *Keras* as the ANN models are built using *keras*. All the scripts used to achieve the objective of this study are uploaded on GitHub (link in Appendix) (Chaturvedi, 2021).

Keras

It is a python library built for making deep learning models as fast and easy for research and practical applications. It can run on top of TensorFlow, powerful Python libraries for fast numerical computing. TensorFlow is a python-friendly end-to-end open-source platform for machine learning. *Keras* provides functional application programming interfaces (APIs) to define complex models, such as multi-output models.

3.4.3 Defining Architecture and Practical Aspects of the Neural Network

Three separate proxy models are built using ANN to predict FOPR, FWIR, and FWPR. Concepts explained in section 2.4.2 of chapter 2 are used to define the architecture of neural networks. The ANN models are built based upon the following steps:

- Data Scaling: Min-Max normalization technique is used to scale the data, recall from section 2.4.2.
- Activation Function: The hidden layers use ReLU activation function (explained in section 2.4.1). The output layer uses linear activation function as model will predict real value output, that is a linear regression problem.
- Loss Function and Metrics: MSE is used to calculate loss function during the training process and used as performance metrics on validation data during HPO.
- Gradient descent and optimizer: Mini-batch gradient descent algorithm with a batch size of 32 is used for HPO. Adam is chosen as the gradient descent optimizer.
- Overfitting prevention: Early stopping is used to prevent the model from becoming too close to the data. "Patience” parameter of 20 is included in the code. This indicates that if the validation error does not decrease after 20 consecutive iterations, the algorithm will

terminate training. Additionally, the dropout technique is used to prevent overfitting. For instance, a dropout value of 0.1 indicates that 10 % of the nodes in each cell state will be randomly dropped out.

- HPO: Four hyperparameters, including learning rate, dropout, layer count, and neuron count, are used in this study. These four hyperparameters are optimized using Bayesian Optimization to determine the combination that produces the slightest error in the validation dataset. Default parameters are fed to the Bayesian optimizer as initial inputs, with total evaluations set to 80. This is accomplished using Python's *skopt* module. Each hyperparameter's starting point and search range used for HPO study are as tabulated in Table 3.6. Ranges of hyperparameters are selected based on empirical analysis. The author started by arbitrarily selecting a range and making changes based on trial-and-error analysis.

Table 3.6: Hyperparameters' starting points and their search space

Parameter	**Starting point**	**Search Space**
Learning rate	0.001	Real Values in [0.0001, 0.01]
Dropout	0.1	Real value in [0.01,0.1]
Layer count	1	Integers in [1,5]
Neuron count	30	Integers in [20,100]

3.5 Evaluating Accuracy of SPMs

MSE, amongst the available regression metrics in Keras, is used to monitor the performance of proxies on validation in HPO study. Besides minimizing the loss function during training and optimizing validation loss to find the best hyperparameters configuration, two more approaches are used to evaluate ”goodness" of the proxy models. The first approach evaluates accuracy of individual SPM, by analyzing MAE, R^2 score, MAPE and NRMSE on training, validation, test samples and blind test data for each proxy. R2 score, MAE and NRMSE are calculated using formulas presented in section 2.4.3. The second approach analyzes NPV crossplot and relative difference in NPV to comment on the combined accuracy of the proxy, and therefore to answer if proxies are accurate enough for optimization study.

After blind test, NPV$_{\text{Eclipse}}$ and NPV$_{\text{SPM}}$ are computed for all the simulation cases used in proxy development process. NPV crossplot is generated, and R^2 score for the trend line is used to comment on combined accuracy of SPMs for optimization study. A good match will indicate that proxies can replicate Eclipse simulator outputs. A strong correlation between the NPV$_{\text{Eclipse}}$

and NPV_{SPM} can be a good indicator for proxies' applicability and performance for optimization study. A weak correlation will suggest a need for improvement in proxy development by retraining proxy models with more samples. Author expects improvement in R^2 score when retraining proxies with more cases.

Relative difference in NPV is computed from Eq. 3.1. A positive relative difference in NPV will indicate that proxies underpredict NPV and vice versa.

$$\text{Relative Difference (\%)} = \frac{NPV_{\text{Eclipse}} - NPV_{\text{SPM}}}{NPV_{\text{Eclipse}}} \times 100 \tag{3.1}$$

In addition to the above two approaches,in second QC, the author observes increment in R^2 score of the NPV crosssplot comparing previously trained proxies with the retrained proxies.

3.6 Building Genetic Algorithm Optimization

This section explains the steps involved in building the genetic algorithm optimizer. Python code for GA is written in the Jupyter Notebook.

3.6.1 Objective Function

In this work, BHPs of producers and injectors are chosen as optimization variable, and NPV as objective function. Detailed discussion on objective function and optimization problem can be found in section 2.2.1. The author have written python code for calculation of $\text{NPV}_{\text{Eclipse}}$ and NPV_{SPM} for one case and group of cases.

3.6.2 Population Size

Population size is the number of individuals in one population. As a rule of thumb, smaller population size is considered to give quicker convergence but with possibility of algorithm getting trapped in local optima. As the total number of control variables is 320 one might expect to require a population size ranging $[320, 2 \times 320]$. But convergence will then be slow, as in GA, it is easy to go through thousands of function evaluations without seeing any improvement and then suddenly see one. Considering **adaptive** nature of proxy, the author tried with small population sizes ranging $[20, 50]$ and observed that smaller number results in faster convergence.

Therefore, four population sizes (20, 30, 36 and 50) are selected to be tested in the sensitivity study.

3.6.3 Selection, Crossover, Mutation, and Termination Operations

In the selection operation, both roulette wheel selection and elitism selection, detailed description available in section 2.2.2, are used. One chromosome with the highest fitness is carried over (elitism selection) to produce the next generation, and the remaining chromosomes are selected using a roulette wheel. Crossover technique calculates the arithmetic mean of both parent's chromosomes. Mutation technique adds a random number in a range of [-20,20] in every gene provided BHP constraints are satisfied. The chosen chromosomes with roulette wheel selection have a chance of doing a crossover and then mutating. In this study, three different probabilities of crossover and three different probabilities of mutation are tested: 25%, 50%, and 75% chance of doing crossover and 25%, 50%, and 75% chance of mutating as stated in Table 3.7. The process is terminated once maximum number of generations are reached. This study uses only 30 generations for sensitivity analysis on GA parameters and 100 generations for final GA run.

Table 3.7: GA parameters for Sensitivity Study

population size	20, 30, 36, 50
mutation probability	25%, 50%, 75%
crossover probability	25%, 50%, 75%

3.7 SPM-GA Optimization

SPMs are coupled with GA optimizer instead of Eclipse simulator, to achieve the main objective of this thesis. Proxy models are loaded outside the optimization loop and used to predict outputs required for NPV calculation for each individual in the population generated during the optimization process. Population is initialized by picking BHPs randomly within the range defined for producer and injector in Table 3.4. After specifying GA control parameters for base run, several combinations of GA parameters presented in Table 3.8 are studied. For each combination, SPM-GA is run five times. Values defined in Table 3.7 result in total of $4 \times 3 \times 3 = 36$ combinations, and $36 \times 5 = 180$ GA runs. To reduce the number of SPM-GA runs, the author first studies effect of population sizes by making in total $5 \times 4 = 20$ runs for four population sizes. Based on the sensitivity study on population size, one cost-efficient population size is chosen for testing combinations of crossover and mutation, hence making $3 \times 3 \times 5 = 45$ more SPM-GA runs. Thus, overall SPM-GA runs are reduced to 65 compared to 180 if all the combinations are to be tested.

$\text{NPV}_{\text{Eclipse}}$ is computed by running the optimal BHP configurations found from 65 optimization runs in Eclipse. The author studies relative difference in NPV from Eq. 3.1 to comment on quality of proxies for optimization study. To comment on the efficiency of the optimizer, increment in NPV is quantified using Eq. 3.2, where $\text{NPV}_{\text{basecase}}$ (i.e., without optimization for base case run presented in Table 3.5) is used as a basis to keep consistency throughout the comparison. Once the desired quality of optimization is achieved, final SPM-GA run is made and NPV increment from Eq. 3.2 is calculated by running optimal BHP settings in Eclipse and obtaining desired outputs for NPV calculation. Higher increment in NPV can be considered outcome of an efficient optimizer.

Table 3.8: Control parameters specified and used for GA algorithm in this work for the first GA run.

Parameter	**Value**
Max Generations	30
Population Size	36
Crossover Probability	0.25
Mutation Strength	0.25
Lower Bound	-20
Upper Bound	20

$$\text{NPV Increment (\%)} = \frac{NPV_{\text{Eclipse}} - NPV_{\text{base case}}}{NPV_{\text{base case}}} \times 100 \qquad (3.2)$$

Chapter 4

Results

This chapter presents the results of well-control optimization study carried out on proxy models coupled with genetic algorithm. It starts by evaluating the performance of each proxy model, then moves on to the sensitivity study of GA parameters, and finally, presents the results of the SPM-GA model.

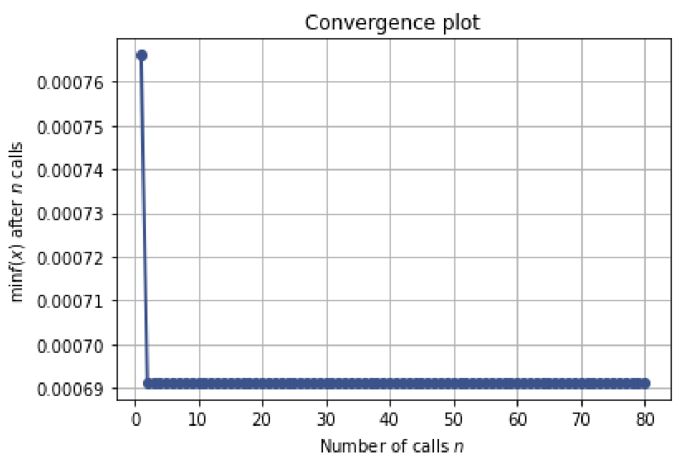
4.1 SPM Results

This section presents the results for three proxy models, each developed for one output parameter (i.e., FOPR, FWIR, and FWPR). It includes convergence plots, optimal hyperparameters and comparison plots for each proxy.

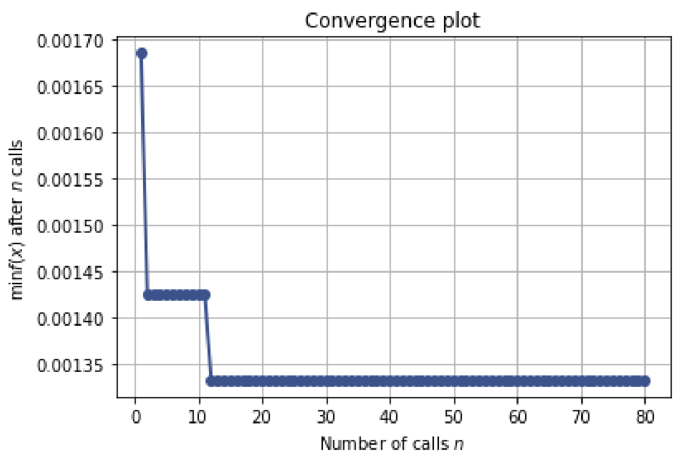
4.1.1 Convergence Plots

Figures 4.1(a)-(c) show convergence plots for three proxy models. The convergence plot for FOPR proxy model in Fig. 4.1(a) shows the surrogate function converged to its minimum in the third evaluation among 80 evaluations. It seems that maximum evaluations can be reduced to save more time. This fast convergence, however, can be limited to this model.

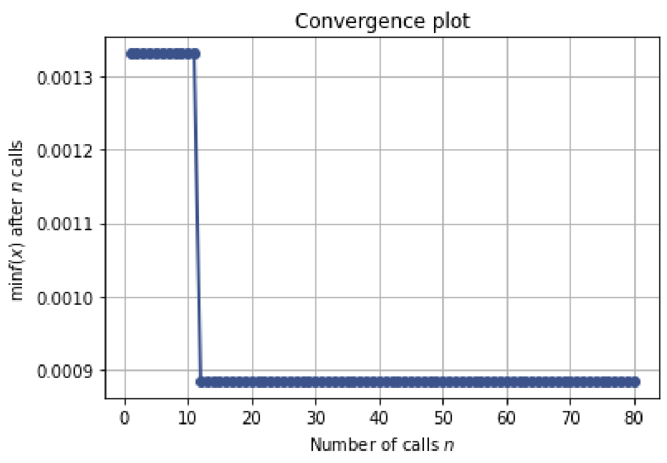
The convergence plot for FWIR proxy model in Fig. 4.1(b) shows that the surrogate function already reached its minimum value at the twelfth evaluation among 80 evaluations. Once again, the HPO converges faster than expected, and total evaluations of 80 appear to be excessive for this task. However, because this faster convergence cannot be applied to another model, there-



(a) FOPR proxy model



(b) FWIR proxy model



(c) FWPR proxy model

Figure 4.1: The convergence plot of the surrogate function in Bayesian optimization for proxy models.

fore, the total evaluation of 80 are retained for the FWPR proxy model. Similar to FWIR proxy model, the convergence plot for FWPR proxy model in Fig. 4.1(c) shows that surrogate function converged to its minimum value in twelfth evaluation among 80 evaluations performed. The Bayesian optimization converges before 15 evaluations for the third time.

Based on these three SPMs, it is possible to conclude that 15 evaluations are sufficient to obtain an optimal combination of hyperparameters. This discovery can be used in the future during retraining proxy models to find the best hyperparameters.

4.1.2 Optimal Hyperparameters

Table 4.1 states the best networks structures that minimize the validation loss for FOPR, FWIR and FWPR proxy models, accomplished after a series of Bayesian optimization on hyperparameters. Variation in the best hyperparameters among three proxies can be an indication of differences in complexity to predict each output.

Table 4.1: The optimum hyperparameters configuration used for training the proxies.

SPM	**Optimum values of Hyperparameters**			
	Learning rate	**Number of layers**	**Neuron count**	**Dropout rate**
FOPR	3.88e-04	3	35	1.17e-02
FWIR	8.5e-04	5	68	0.01
FWPR	3.88e-04	2	98	1.16e-02

A similar learning rate of $3.88e-04$ is observed for FOPR and FWPR proxy, while FWIR proxy works best with the learning rate of $8.5e-04$. FWIR proxy consists of five hidden layers, highest among the three proxies, while FWPR and FOPR proxy have only 2 and 3 hidden layers, respectively. In contrast to hidden layers, neurons count per layer is highest (98 nodes) for FWPR proxy, followed by 68 for FWIR proxy and lowest neuron counts of 35 for FOPR proxy. This could mean that it is relatively easier to predict FOPR since the model needs fewer neurons. Overall, from hyperparameter combinations obtained for the three proxies, it can be concluded that FWIR and FWPR behavior is more complicated to learn and, therefore, requires heavy architecture.

4.1.3 Performance Metrics

Table 4.2 presents values of three performance metrics calculated for training, validation and test samples. Best R^2 of 0.99, obtained for all the samples for FOPR and FWPR proxy, shows a strong correlation between predictions of proxy models and targeted outputs. In contrast, variation in R^2 is observed among different samples for FWIR proxy, 0.98 on training, 0.94 on validation and 0.97 on test data. This variation implies that good correlation is observed on training and test samples, while relatively lower R^2 score on validation data is result of some deviations in FWIR predictions for validation data.

Table 4.2: Three performance metrics for each proxy.

SPM	R^2			MAE [Sm^3/day]			NRMSE [%]		
	Training	Validation	Test	Training	Validation	Test	Training	Validation	Test
FOPR	0.99	0.99	0.99	7.3	7.7	8	5.7	6.7	6.8
FWIR	0.98	0.94	0.97	9.4	9.0	8.9	3.1	2.9	2.5
FWPR	0.99	0.99	0.99	8.9	8.8	9.3	4.7	4.4	4.3

Overall, all the proxies show acceptable mean absolute errors, ranging from 7.3 to 9.3 Sm^3/day, across different datasets. Considering difference in numerical ranges of FOPR, FWIR, and FWPR, NRMSE is calculated to draw comparison among three proxies. FOPR yields highest, followed by FWPR, and FWIR yields lowest NRMSE among the three proxies across all the samples. Possible reasoning for lowest NRMSE for FWIR model could be a larger mean of targeted FWIR.

Acceptable ranges of R^2, MAE and NRMSE for all the proxy models indicate that proxies are deemed fit to predict the desired outputs. Results of blind test will further confirm this.

4.1.4 Blind Test

The trained proxy models are validated against completely blind samples, refer to section 2.4.2 for definition. Figures 4.2-4.4 display blind test results for the three proxy models, comparing proxy responses with the simulation outputs. In the plots, results generated by SPMs are shown by dashed lines, while results from Eclipse simulator are shown by solid lines.

As demonstrated in the results, the time-dependent proxy models have a reliable ability to predict different desired outputs, as very close match between the proxy emulated outputs and those of the Eclipse simulator are noticed. In addition, Table 4.3 tabulates different performance metrics calculated for blind test.

It should be noted that a high R^2 score is achieved for all the proxies. NRMSE values are also within the range as observed on training/validation/test samples, presented in Table 4.2. According to Table 4.3 and Figs. 4.2-4.4, the developed proxies are deemed satisfactory in approximating Eclipse simulator outputs.

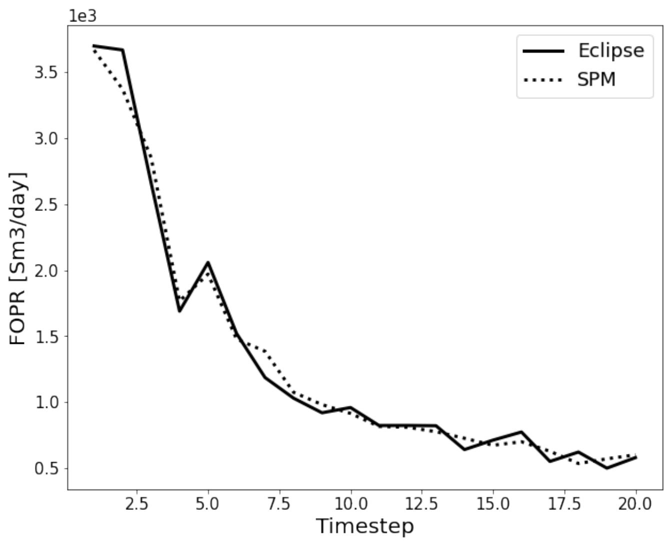


Figure 4.2: The comparison between FOPR_{SPM} and $\text{FOPR}_{\text{Eclipse}}$ for blind test dataset

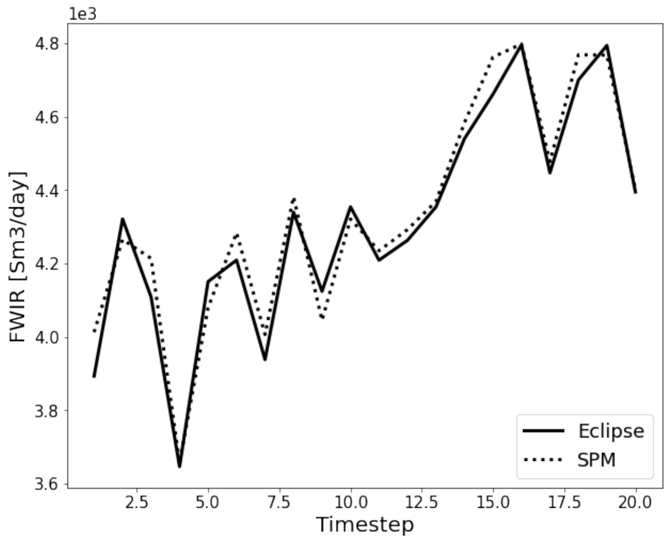


Figure 4.3: The comparison between FWIR_{SPM} and $\text{FWIR}_{\text{Eclipse}}$ for blind test dataset

Table 4.3: The blind testing results for each proxy.

SPM	**Blind Test**		
	R^2	**MAE [Sm^3/day]**	**NRMSE [%]**
FOPR	0.98	9.7	8.1
FWIR	0.96	9.3	2.5
FWPR	0.98	10.6	5.8

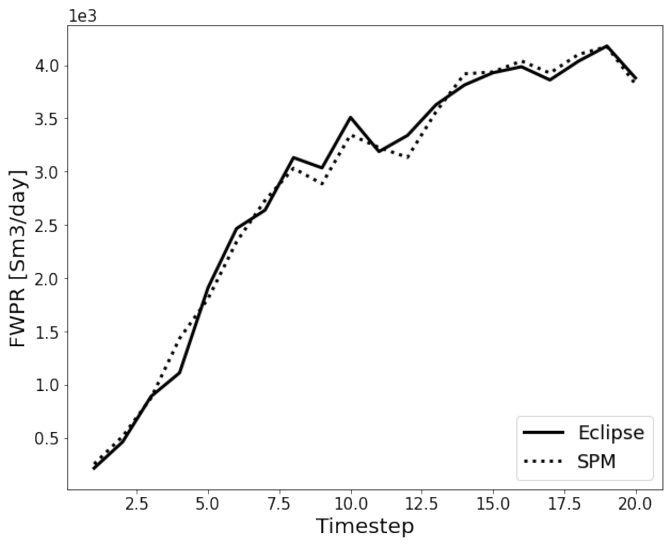


Figure 4.4: The comparison between $FWPR_{SPM}$ and $FWPR_{Eclipse}$ for blind test dataset

4.1.5 Combined Accuracy of SPMs

From the results discussed in previous two subsections, either in training/validation/test or in blind test, individual prediction capabilities of the established proxy models are documented. Before proxy models can be used for optimization study, the combined accuracy of these proxies should be studied. Relative differences in NPV and NPV crossplot are used to check the combined accuracy of proxy models.

Relative Difference in NPV

To comment on the combined accuracy of the three proxies, NPV is calculated from Eq. 2.1 using proxy emulated outputs and outputs from Eclipse. Relative difference in NPV is then calculated for the base case and blind test. Table 4.4 shows small difference between $NPV_{Eclipse}$ and NPV_{SPM}, that is also confirmed by low relative difference of 2.76% and 4.52% calculated using Eq. 3.1. This small relative difference gives confidence in the deployment of proxy models for optimization task.

Table 4.4: NPV for base case and blind test dataset.

Case	**$NPV_{Eclipse}$ [billion USD]**	**NPV_{SPM} [billion USD]**	**Relative Difference [%]**
Base case	1.60	1.53	4.52
Blind Test	2.09	2.03	2.76

NPV Crossplot

To illustrate the combined performance of proxy models $NPV_{Eclipse}$ and NPV_{SPM} are calculated for the existing 85 Eclipse runs used for proxy development process. Figure 4.5 shows correlation between $NPV_{Eclipse}$ and NPV_{SPM} for these cases. R^2 score of 0.96 shows a good correlation and provides confidence in deployment of proxies for optimization studies.

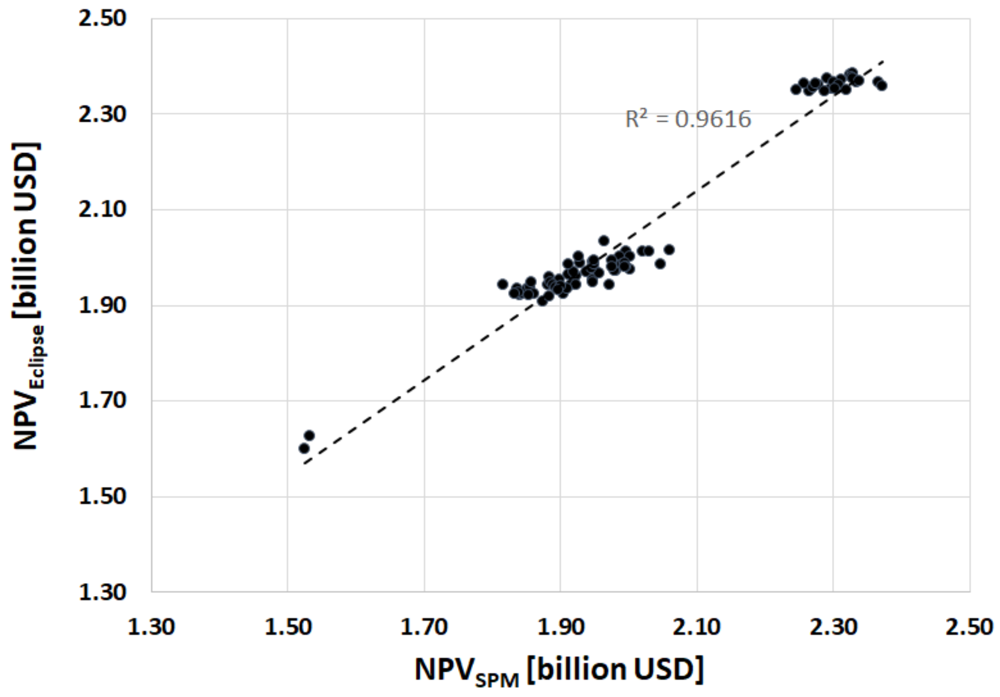


Figure 4.5: Correlation between NPV_{SPM} and $NPV_{Eclipse}$ for 85 cases used for building proxies.

This R^2 score for the NPV crossplot will be used later to check if retraining is required and to compare different versions of each proxy model.

4.2 SPM-GA Optimization Results

In the last stage, to accomplish the objective of this study and solve well control optimization problem formulated in section 3.3, the dynamic proxy models are coupled with GA. This section presents the results obtained from SPM-GA coupling, starting from sensitivity analysis of GA parameters and finding the best BHP configuration.

4.2.1 Sensitivity Analyses

Population Size

Four population sizes as shown in Table 4.5 are tested to determine the cost-efficient population size for the optimization run, other parameters kept as defined in Table 3.8. Figure 4.6 displays that population size of 50 yields better NPV at very first generation, but experiences slow increment in the subsequent generations. It also displays that population size of 20 yields highest NPV at 30th generation.

Table 4.5: Results for the best NPV (SPM-GA) obtained from five runs for four population sizes keeping other configuration settings intact.

Population Size [-]	**Max NPV_{SPM}** [billion \$]	**Mean NPV_{SPM}** [billion \$]	**Worst NPV_{SPM}** [billion \$]	**Standard Deviation** [billion \$]	**Elapsed Time** [HH:MM:SS]
20	2.17	2.10	2.07	0.042	00:40:48
30	2.10	2.06	2.01	0.029	00:55:51
36	2.11	2.08	2.07	0.026	01:12:10
50	2.09	2.05	2.03	0.041	03:18:36

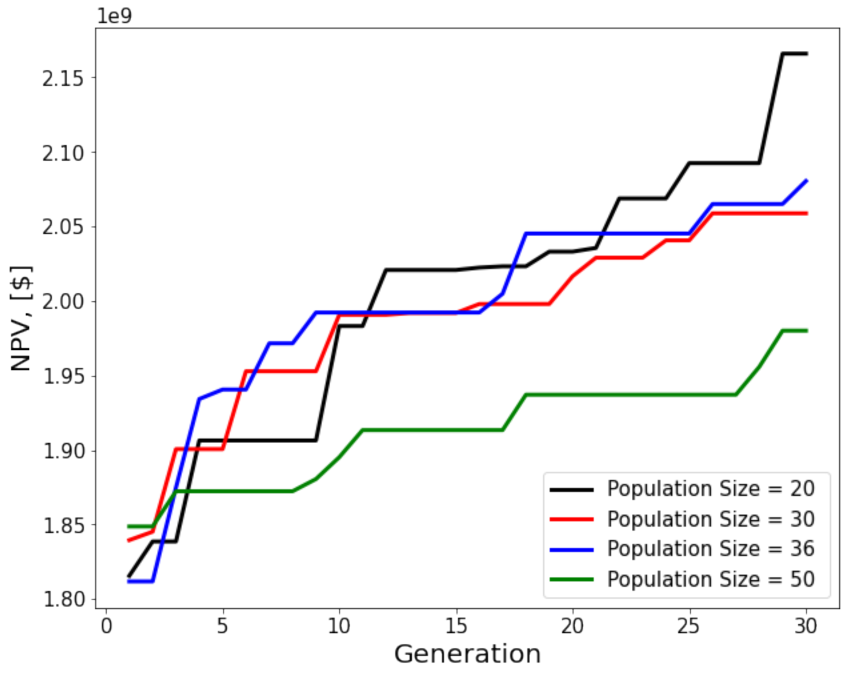


Figure 4.6: Comparison of NPV increase with generations for various population sizes for one of five optimization runs performed

In general, increasing the population size increases the likelihood of finding a better solution in the next generation, but it also increases computational time. Therefore, both the optimization results and elapsed time of each population size are recorded. Table 4.5 presents elapsed time and optimized NPV for each setting. According to Fig. 4.6 and Table 4.5, it is observed that population size of 20 performs best when maximum generations are set to 30, yields highest NPV of 2.17 billion USD and takes least computational time to run. Therefore, population size of 20 is the most cost-efficient one for this problem and is used for subsequent analysis.

Crossover Probability and Mutation

A total of 9 possible combinations of crossover and mutation probability (25%, 50%, and 75%) are tested. Each combination consists of similar population size of 20, maximum generations of 30, and remaining parameters as defined for base run in Table 3.8. Figure 4.7 illustrates NPV increase with generations for one of optimization runs amongst the five runs performed for all 9 combinations.

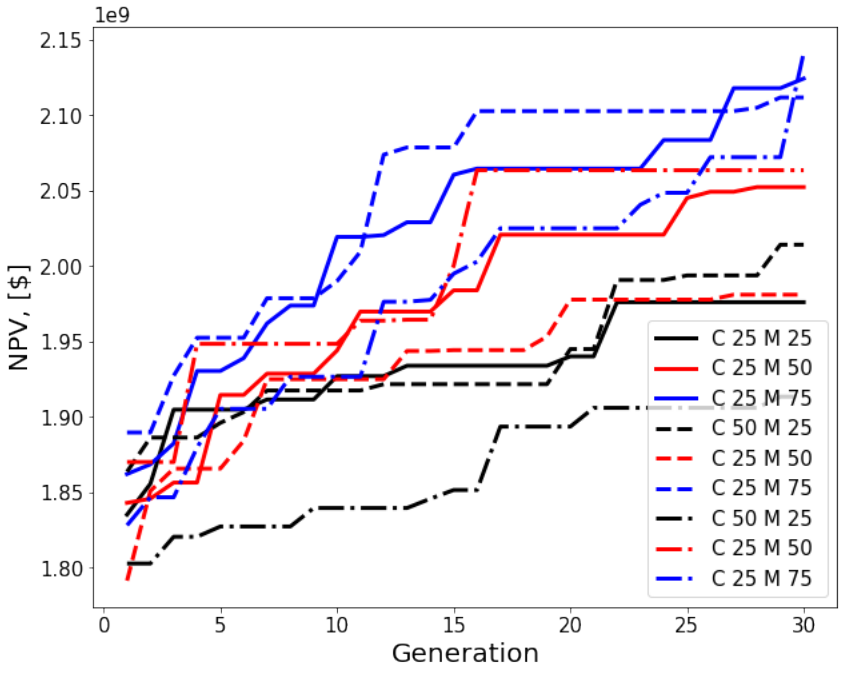


Figure 4.7: Comparison of NPV increase with generations for one of five optimization runs performed for combinations of mutation and crossover probability (C for Crossover and M for Mutation)

Table 4.6 shows the maximum, mean and minimum NPV_{SPM} value among five runs for each combination. Standard deviation among the five runs presented in Table 4.6 shows variation among different combinations. As observed, crossover probability of 25% and mutation probability of 75%, yield highest value of max, min and mean of NPV_{SPM}. The elapsed time is not

Table 4.6: Average maximum NPV for every combination of crossover and mutation probability keeping other configuration settings intact.

Crossover Probability [%]	Mutation Probability [%]	NPV_{SPM} [billion USD]			
		Max	**Mean**	**Min**	**Std Dev**
25	25	2.04	2.02	1.98	2.78e-02
25	50	2.13	2.08	2.01	4.23e-02
25	75	2.19	2.14	2.11	2.85e-02
50	25	2.07	2.00	1.95	4.23e-02
50	50	2.07	2.04	1.98	3.67e-02
50	75	2.11	2.07	2.03	2.61e-02
75	25	2.00	1.97	1.91	3.61e-02
75	50	2.06	2.02	1.95	4.10e-02
75	75	2.16	2.11	2.07	3.43e-02

included in Table 4.6, as it was about 45 minutes for all the combinations. Reason for similar elapsed time is the same population size for all the combinations.

From Table 4.6, it is noticed that higher mutation probability performs better than higher crossover probability in all the cases. Possible reason could be that high mutation rate results in faster convergence by introducing diversity in the population. In contrast, higher crossover probability performs worse. The reason could be that a high crossover probability may introduce too many new variables into the population, causing the search to become longer or worse. According to Fig 4.7 and Table 4.6, crossover and mutation probability of 25% and 75%, respectively, prove to be the best combination for this optimization study.

Figure 4.10 shows R^2 of 0.87 for NPV crossplot generated using 150 (85+65) runs for current proxies (shown by blue circle). This decrease in R^2 from previous value of 0.96 (refer to Fig. 4.5 which was generated for 85 cases), indicates the potential that the combined accuracy of proxy models can be improved by retraining the proxy models.

4.2.2 Quality Checking the SPMs for Optimization Task

Optimal BHP settings obtained from 65 optimization runs for GA sensitivity analyses are run with Eclipse. NPV_{SPM} and $NPV_{Eclipse}$ are calculated from Eq. 2.1 using proxy emulated outputs and outputs from Eclipse, respectively. Relative difference in NPV is then calculated for whole 65 cases. Table 4.7 shows maximum, minimum, mean and standard deviation for absolute values of relative difference in NPV. The average absolute relative difference in NPV of 2.75% can be considered satisfactory, which shows good performance of proxy models for the purpose of study. However, Fig. 4.8 displays higher variation in relative difference in NPV among 65 cases, ranging from -8.15% to 9.36%. Hence, there is potential to improve the performance of proxy models.

Table 4.7: Min, max, mean and standard deviation for absolute values of relative difference in NPV, based on 65 runs.

Number of samples	Relative Difference in NPV [%]			
	Max	Mean	Min	Std Dev
60	9.36	2.74	0.01	2.13

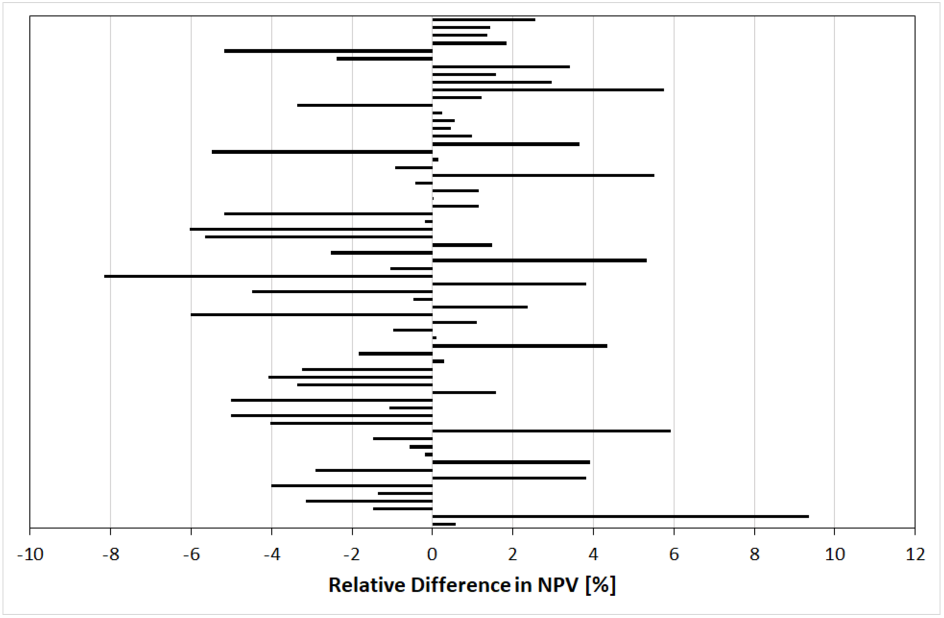
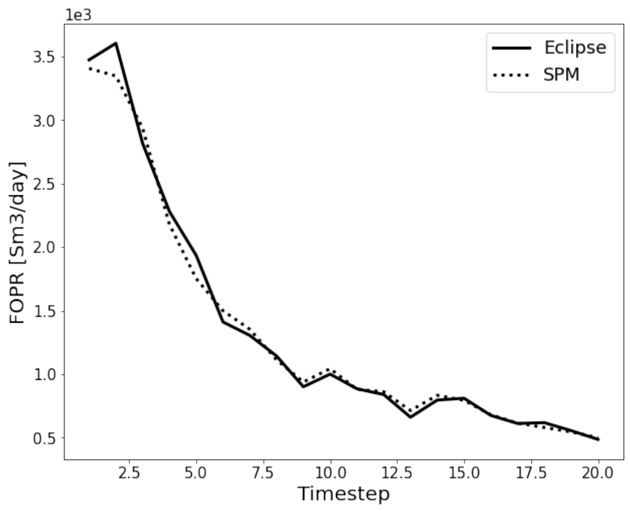


Figure 4.8: Relative difference in NPV observed for 65 runs from GA sensitivity analysis.

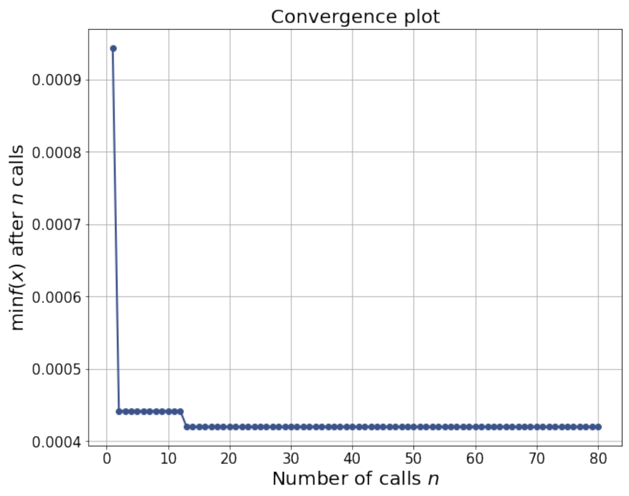
4.3 Retraining SPMs

Proxies are retrained by including data generated from 65 Eclipse runs performed on GA sensitivity results. This is done to improve the quality of proxies and in a way making proxies adaptive. This process took around 15 minutes for each proxy, and once all the proxies are trained, blind validation is performed on blind runs and results are presented in Fig. 4.9 for all the proxies.

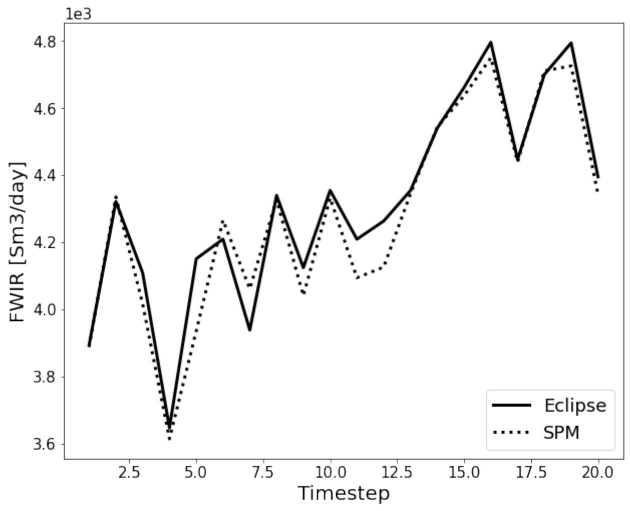
Table 4.8 states the best networks structures that minimize the validation loss for FOPR, FWIR and FWPR proxy models, accomplished after a series of Bayesian optimization on hyperparameters. ANNs architecture for the retained proxy models differs from those of the previous set of



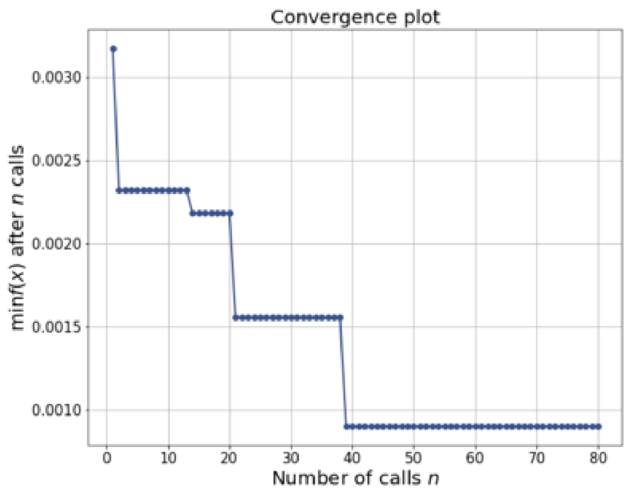
(a) Blind test performance of FOPR proxy



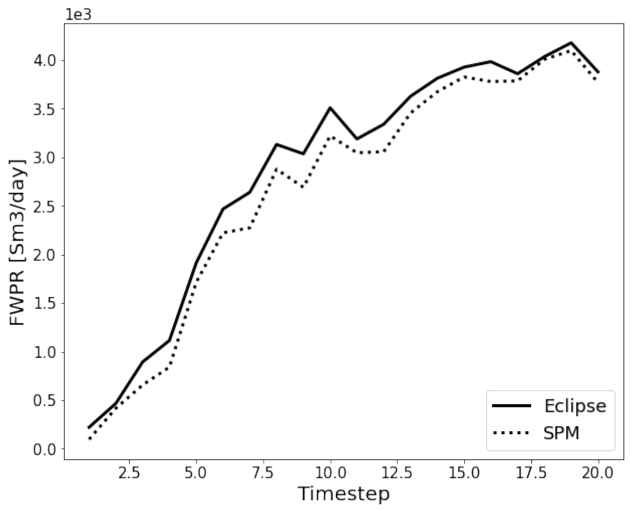
(b) Convergence plot for FOPR proxy



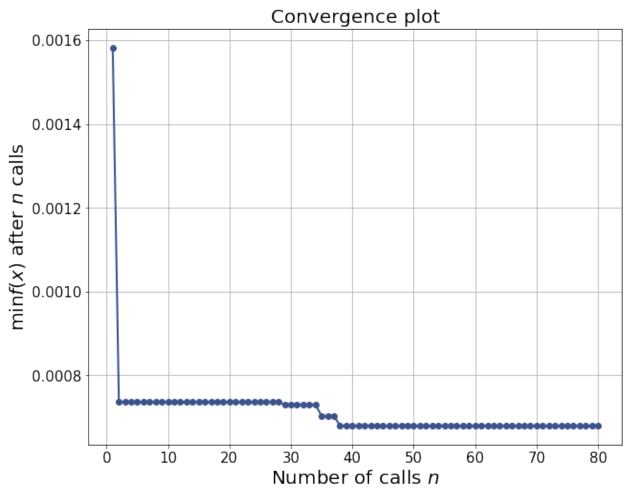
(c) Blind test performance of FWIR proxy



(d) Convergence plot for FWIR proxy



(e) Blind test performance of FWPR proxy



(f) Convergence plot for FWPR proxy

Figure 4.9: Visualizations of blind test results and convergence plots for three proxies

proxy models presented in Table 4.1. It could be because the increasing number of samples also increases complexity in the behavior to be learned by ANN. FOPR and FWPR proxy models have similar best hyperparameters, indicating similar complexity to predict FOPR and FWPR. In contrast, a denser architecture for FWIR proxy indicates more complexity to learn FWIR behavior.

Table 4.8: The optimum hyperparameter configuration used for retraining the proxies.

SPM	**Optimum values of hyperparameters**			
	Learning rate	**Number of layers**	**Neuron count**	**Dropout rate**
FOPR	3.9e-02	3	35	1.17e-02
FWIR	1.0e-02	5	43	6.89e-02
FWPR	3.9e-02	3	35	1.16e-02

Table 4.9 tabulates the performance metrics of proxy models on training, validation, and test samples to check predictability of proxies. Based on the results presented, it is observed that proxy models show a strong correlation on different datasets, except for R^2 of 0.93 observed for FWIR proxy on validation dataset, in agreement with 0.94 observed for previous version (refer to Table 4.2). According to Figs. 4.9 (a, c, e) and Tables 4.9 and 4.10, the updated proxy models can predict satisfactory outputs.

Table 4.9: Performance metrics for the retrained SPMs.

SPM	**R^2**			**MAE [Sm^3/day]**			**NRMSE [%]**		
	Training	**Validation**	**Test**	**Training**	**Validation**	**Test**	**Training**	**Validation**	**Test**
FOPR	0.99	0.99	0.98	7.9	8.5	8.7	6.6	7.8	8.9
FWIR	0.99	0.93	0.95	7.4	8.9	8.7	2.0	2.8	2.7
FWPR	0.99	0.99	0.99	10.5	9.5	10	6.1	5.0	5.6

Table 4.10: Blind testing results for the retained SPMs.

SPM	**Blind Test**		
	R^2	**MAE [Sm^3/day]**	**NRMSE [%]**
FOPR	0.99	7.6	6.46
FWIR	0.93	7.5	1.83
FWPR	0.97	13.6	7.3

Comparing Previous SPMs with Retrained SPMs

NPV crossplots are used to compare the combined accuracy of both versions of proxies before deploying the retrained proxies for rest of the study. Figure 4.10 shows correlation between $NPV_{Eclipse}$ and NPV_{SPM} for 150 cases, for both versions. Recall that retrained versions are built using 150 cases, while the previous versions were built based on 85 cases. Improvement in correlation coefficient is observed in Fig. 4.10, from 0.87 for previous proxies to 0.91 for retrained proxies. This provides confidence in deployment of the retrained proxies for further optimization studies.

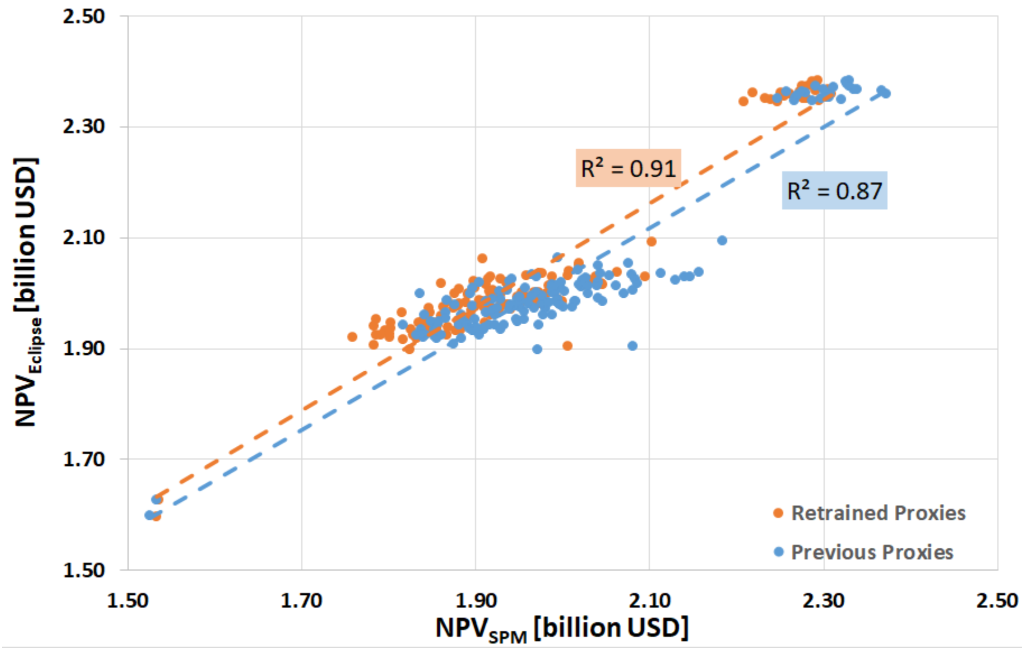


Figure 4.10: Comparison of NPV crossplots on 150 cases for two versions of proxies

4.4 Best Case

After retraining the proxies and deciding upon the best optimization configuration settings for GA, which are population size of 20, mutation probability of 75%, crossover probability of 25% and keeping other parameters unchanged as shown in Table 3.8, GA optimization is run until 100 generations to find the maximum NPV. Figure 4.11 depicts NPV_{SPM} development over 100 generations.

After 100 generations, the maximum NPV achieved is 2.188 billion USD. However, this result should be confirmed with reservoir simulator. The BHP configuration that yields maximum

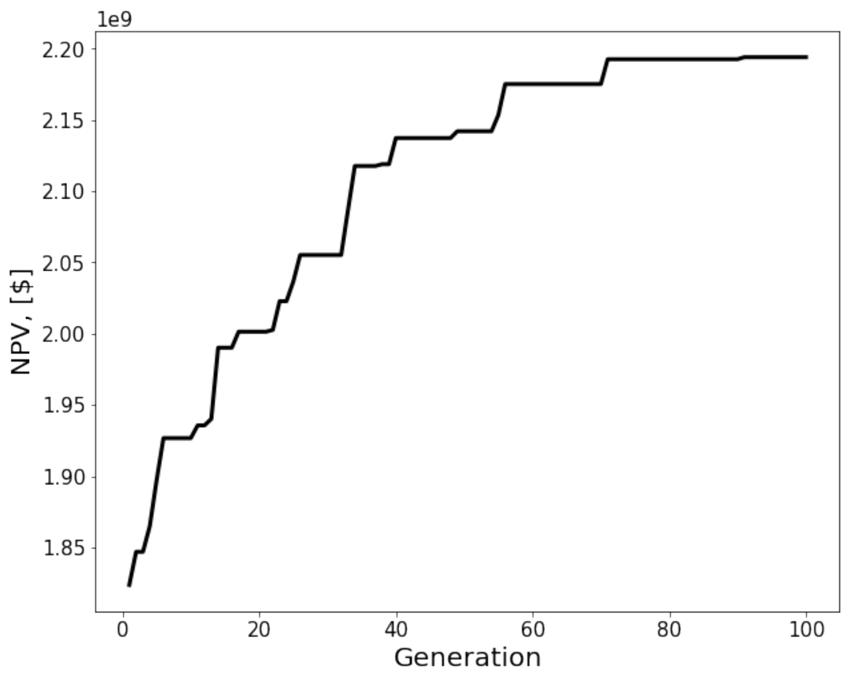


Figure 4.11: NPV improvement for optimization run with population size of 20, mutation probability of 75% and crossover probability of 25% for 100 generations.

NPV_{SPM} is run on Eclipse and results are presented in Appendix. The time elapsed to run this optimization is **1 hour 42 minutes 50 seconds**.

$NPV_{Eclipse}$ resulted in 2.170 billion USD, or 17.5 million USD lower than NPV_{SPM}. Relative difference between $NPV_{Eclipse}$ and NPV_{SPM} is only **0.81%**, which represents a small error. This difference is even smaller than that for blind test data and base case, discussed in Table 4.4. Optimum $NPV_{Eclipse}$ of 2.17 billion USD is a 33% increase from the base case's $NPV_{Eclipse}$ of 1.60 billion USD. This increase in NPV demonstrates the genetic algorithm's ability as a useful optimization algorithm for field development optimization problems. This increase in NPV was also achieved by using reliable proxies to evaluate the objective function.

The BHPs for best case are within the bounds for the majority of wells, as tabulated in Table A.1. All producers and injectors experience variation in BHP throughout the production life of the field. Figures A.2 compares FOPT, FWPT and FWIT profiles for base case and best case.

4.5 Computation Time and Resources

The author performed the same well control optimization on Maur cluster using Eclipse simulator and FieldOpt software framework in the specialization project (Chaturvedi, 2020). Maur cluster is a linux cluster mainly funded by the department of Geosciences and Petroleum (NTNU) ($21 \times DellR730$), with additional nodes ($7 \times Dell$ R620) from EPT and later upgraded somewhat by NTNU IT. The Dell R730 nodes also have 2× Nvidia Titan X graphics cards used for calculations. For one PSO run, with swarm size of 24 and maximum generations of 30, elapsed time for one optimization run was around **15 hours 30 minutes** (considering the situations with availability of all the required nodes) when 25 Eclipse licenses were used for parallel computing. Moreover, 16.8 GB of storage space was required to save the outputs from each optimization run.

In contrast to specialization project, this study was carried out on a Machine with Intel(R) Core(TM) i5-8300H CPU 2.30GHz with 8 GB RAM and required only one Eclipse license for database preparation. Produced Keras models take up 305 KB storage space, and around 160 Eclipse simulations take 4.24 GB of space. Time taken for one NPV calculation which requires predicting three outputs from three proxy model, is **0.13 seconds**. Overall, SPM-GA model takes around **43 minutes** for one GA run with a population size of 20 and maximum generations of 30. Computational time has been reduced by 95%. Storage space requirement of the SPM-GA model and Eclipse simulations for database preparation are insignificant compared to 16.8 GB of storage space requirement for just one optimization run with FieldOpt and Eclipse.

Chapter 5

Discussion

The results presented in previous chapter are discussed to provide insights required to achieve the objectives of this study.

5.1 Interpretations and Implications of Results

Despite the high non-linearity of the optimization problem solved in this work, proxy models proved to be effective in achieving the main objective of this study.

5.1.1 Constructed SPMs

1. Based on the increment obtained in NPV, field-based proxies, i.e. proxies to estimate FOPR, FWIR, and FWPR, proved sufficient for the optimization study done in this project. Furthermore, even though the influence of previous timesteps on current timestep output was not considered when training the ANNs, this study discovered that ANNs do an excellent job of building a black box to predict the dynamic reservoir behavior.

2. The proxy models are objective specific and can only be applied to the task for which they were designed; therefore, a proxy can be considered reliable if it fulfils the objective it was developed for. Hence, the established proxy models are reliable as they fulfill the main objective of the study, i.e., finding BHP settings that maximize NPV.

3. The development of a proxy model for well control optimization necessitates knowledge of both AI based technologies and Petroleum Engineering. Furthermore, deciding on in-

put parameters for establishing these proxies necessitates a high level of understanding in terms of possible optimization problem formulations.

4. The differences in optimal hyperparameters between the three proxies may reflect differences in the complexity of FOPR, FWIR, and FWPR behavior. Overall, based on the hyperparameter combinations obtained for the three proxies, it is possible to conclude that FWIR and FWPR behavior is more challenging to learn and, as a result, requires heavy architecture.

5. Lower NRMSE values in validation and test data compared to training data indicate that FWIR and FWPR proxies predict satisfactorily for the data they never learned.

6. It is important to note that the complexity of the reservoir has no significant effect on the time it takes to run proxy models. Methodology proposed in this study was tested on a complex synthetic model (with $192,750$ active grid cells and heterogeneous reservoir), The minimum number of evaluations required for the Bayesian optimization study for HPO experienced an increase with the number of data samples. Therefore, the number of data samples used to construct the proxies influence the training time.

5.1.2 SPM-GA Optimization

The foundation for adaptive proxy modeling presented in this work can be a basis to develop an automated workflow for proxy building. The idea is to retrain the proxies during the optimization process. Adaptive proxy models will improve the applicability of proxy models for optimization. Automating the process will save time and improve efficiency. Other discussion based on SPM-GA optimization are as follows:

1. Finding an optimal solution to complex, high-dimensional problems frequently necessitate the use of very expensive objective function evaluations. For example, one objective function evaluation problem may take several hours or several days. In this case, it may be better to forego the exact evaluation and instead rely on approximated fitness. SPM-GA coupling does an excellent job to achieve significant reduction in run time, computational resources, and storage space, as presented in Chapter 4.

2. It can be noted that SPM needs not to be 100% accurate but accurate enough for optimization purposes. Finding measures to quality check usefulness of the proxies is an important task to consider. Three performance metrics used in this study showed consistency among different samples.

3. Crossplot between NPV_{SPM} and $\text{NPV}_{\text{Eclipse}}$ proved to be a reliable measure to comment on combined accuracy of SPMs for optimization study. But the main question that need to be answered is how accurate SPMs need to be for optimization process. If the problem is complex and has erratic behavior or highly non-linear, accuracy is of utmost importance, but simpler smooth problem does not require highly accurate SPMs for optimization work.

4. Data used for development of proxy development includes water breakthrough at initial timesteps. Therefore, it resulted in the optimal solutions with water breakthrough observed in the early life of the Field.

5. It is essential to note that BHP range is chosen in a way to avoid the cases when reservoir pressure goes above injector BHPs or below the producer's BHP and results in shut off the well. Such behavior is not included in the proxy development. However, within specified region, SPMs are extremely useful, based on the results presented in previous chapter.

5.2 Limitations

Although the established SPMs and proposed SPM-GA model appear promising, there are some limitations to consider for application in decision making.

5.2.1 Constructed SPMs

1. The main limitation of the established proxy models is that they can only be applied to the reservoir used for database preparation in developing the models. In addition, SPMs are only trained for a single reservoir condition. Therefore, any changes made to the reservoir, such as adding a new well, changing position of wells, production scheduling or addition of EOR, would require redevelopment of SPMs, hence, making the existing SPMs useless.

2. Preparation of dataset requires some planning and time, in addition to simulation time. For this model, one simulation took around 30 minutes to run on a simple machine. Therefore, spending a considerable amount of time preparing a dataset by running hundreds of simulation runs can be cumbersome if the quality of proxies is not good. Therefore, this study used 30 samples initially and increased the number of samples to 85, which is much less compared to total number of Eclipse simulations a traditional optimization would need.

3. Currently, the closest thing to estimating real reservoir behavior is a reservoir simulation model. As the presented proxies are built based on reservoir simulator data, they can be at most as accurate as the simulator. The outcome of a reservoir simulation may not be replicated in a real-world application. Hence, proxies based on an inaccurate reservoir model would produce inaccurate results.

4. Another observation is that training the SPM can take a significant amount of time. For this project, it took around 10 to 15 minutes to train each SPM, and most of this time was spent on hyperparameter optimization. Moreover, when adding more dataset into training the proxies, hyperparameter tuning results in different settings, so it is wise to run hyperparameter optimization whenever proxy is retrained.

5.2.2 SPM-GA Optimization

1. The optimization study is approached only using one optimization algorithm, i.e. Genetic Algorithm. When doing optimization, especially with proxy models, it is better to work with two or more optimization algorithms to confirm the obtained result and to check if the obtained optimal solution is global or local optimum.

2. Well control optimization problem, which is the main objective of current study, can be improved. Rather than optimizing BHPs of all the wells, rates of injectors and BHPs of producers can be optimized. This will also influence the datasets required for proxy building and other aspects of the study.

3. Randomness in GA algorithm makes it challenging to perform sensitivity analysis. Making five runs for each GA configuration provides some confidence level, but variation in results among five runs needs to be considered in decision making. In this study, an acceptable standard deviation among five GA runs is obtained. However, five SPM-GA runs lead to different optimal BHP configurations, indicating that the optimal solution is not converging to global optimum.

4. Goodwin (2018) discussed that the danger associated with any optimization algorithm is that one explores only on a small dimensional hyperplane, and never explores orthogonal to the hyperplane - moving around hyperplane one only get to explore a tiny fraction of the total space. Here, population size of 20 is used, which is a small number compared to the dimension of the optimization problem, i.e. 320.

5. This study uses relative difference in NPV to comment on the combined ability of the proxies for optimization task. However, one should also consider other possible measures,

for example, testing the relationship "Eclipse predicted NPV increase" vs "SPM predicted NPV increase" from one iteration to the next to get a sense about the current validity of the proxies.

6. The oil price chosen here is representative of current market. Considering the dramatic change to the oil industry and the world recently, sensitivity of oil price on NPV should be considered.

Chapter 6

Conclusions and Recommendations for Further Work

6.1 Conclusions

This study has been done based on the objectives stated in section 1.2 to obtain the results presented in Chapter 4. The author applied a methodology to develop three ANNs to serve as dynamic proxy models for well control optimization in a synthetic field, named Olympus. Eighty-five runs were designed to train these proxy models, and five extra blind runs were employed for performance check and blind validation. After checking accuracy and robustness of each proxy model, they were used as substitutes of numerical reservoir simulator in optimization process. As anticipated, proxy modeling based on ANN proved to be a low-cost alternative to reservoir simulation, particularly for well control optimization problem solved in this study.

To begin, SPM-GA coupling was used to study the sensitivity of GA parameters, and thirteen combinations were tested, with five optimization runs for each combination. BHP configurations from all GA sensitivity runs were run in numerical simulator and NPV calculated from simulator's outputs were compared against NPV calculated from predictions of proxy models. Proxy models were retrained with 85 (previously designed) plus 65 (new) cases, followed by blind validation and comparing results for NPV crossplot. In comparison to previously trained proxy models, the increase in R^2 score of the NPV crossplot for 150 samples confirmed the applicability and robustness of the retrained proxy models. Retrained proxies were integrated with GA and then optimum well control conditions (including BHPs of ten producers and six injectors for twenty timesteps) were obtained using SPM-GA coupling. Integration of proxy models with genetic algorithm proved to be successful as 35% increment in NPV was observed for the synthetic model tested. Highest NPV obtained from SPM-GA was 2.19 billion USD and corresponding $NPV_{Eclipse}$ was 2.17 billion USD. Results presented in Chapter 4 prove a significant reduction in running time and space, without sacrificing the accuracy in optimization process. The established proxy models are adaptive. From the discussion, other conclusions of this study can be drawn as:

- The author devised and tested a systematic workflow to develop adaptive proxy models for well control optimization study in a synthetic field.
- Results show that three proxies are enough to learn the behavior of dynamic data required for well control optimization study. ANN architecture for some proxies is not that complex, in comparison with the complexity and non-linearity of the flow behavior proxy models are trained to replicate. 5 layers and increasing number of nodes results in a better model and does not need additional computational time to run these models.
- Eighty-five data samples proved to be enough for optimization problem solved in this work, however, leveraging the new data generated from sensitivity analysis for retraining the proxy models, has improved performance of SPM-GA coupling.
- As expected, maximum reduction of 95% in computation time was observed for one optimization run.

6.2 Recommendations for Further Work

The thesis is focused on the author developing a methodology for building SPM-GA model for well control optimization of a synthetic model, that can be applicable to fields on NCS. There are some recommendations to consider for future work.

1. Given the time constraints for this thesis, the proposed workflow is only tested on one realization of Olympus model. It would be interesting to test the established methodology on a real field case to check its applicability in the oil and gas industry.
2. Given the limitation of genetic algorithm, such as slow convergence issues, one could try other optimization algorithms such as particle swarm optimization and asynchronous parallel pattern search to test if shortcomings of GA could be addressed and in addition, check if this workflow could be generalized to any optimization algorithm.

3. The SPM-GA model is developed and designed using only one programming framework, and it should be interesting to compare performance with other Machine Learning libraries and toolboxes. The Matlab-based Machine Learning toolbox is worth considering and can be compared to the other approaches. Integration of proxies with existing software framework (such as FieldOpt) for optimization could be another possibility to investigate.

4. For further work it would be interesting to train a model that can predict NPV directly for a wide range of BHPs. It would be challenging and interesting to train a model for a large range of BHPs.

References

Amini, S., 2015. Developing a Grid-Based Surrogate Reservoir Model Using Artificial Intelligence. Graduate Theses URL: https://researchrepository.wvu.edu/etd/5096, doi:10.33915/etd.5096.

Arouri, Y., Sayyafzadeh, M., 2020. An accelerated gradient algorithm for well control optimization. Journal of Petroleum Science and Engineering 190, 106872. URL: http://www.sciencedirect.com/science/article/pii/S0920410519312884, doi:10.1016/j.petrol.2019.106872.

Awotunde, A., Sibaweihi, N., 2018. A multi-level clustering method for field development optimization under uncertainty, in: EAGE-TNO Workshop on OLYMPUS Field Development Optimization, European Association of Geoscientists and Engineers, EAGE, Barcelona, Spain.

Awotunde, A.A., 2019. A Comprehensive Evaluation of Dimension-Reduction Approaches in Optimization of Well Rates. SPE Journal 24, 912–950. URL: https://doi.org/10.2118/194510-PA.

Badru, O., Kabir, C.S., 2003. Well Placement Optimization in Field Development, Society of Petroleum Engineers. URL: https://www.onepetro.org/conference-paper/SPE-84191-MS, doi:10.2118/84191-MS.

Baumann, E.J.M., 2015. FieldOpt: Enhanced Software Framework for Petroleum Field Optimization - Development of Software Support System for the Integration of Oil Production Problems with Optimization Methodology. 82 URL: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2351001. publisher: NTNU.

Baumann, E.J.M., Dale, S.I., Bellout, M.C., 2020. FieldOpt: A powerful and effective programming framework tailored for field development optimization. Computers & Geosciences 135, 104379. URL: http://www.sciencedirect.com/science/article/pii/S0098300419301013, doi:10.1016/j.cageo.2019.104379.

Bellout, M.C., 2014. Joint Optimization of Well Placement and Controls for Petroleum Field Development. NTNU. URL: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/280037. iSSN: 1503-8181.

Bellout, M.C.R., Volkov, O., 2018. Development Of Efficient Constraint-Handling Approaches For Well Placement Optimization URL: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2596882. publisher: European Association of Geoscientists and Engineers.

Cavazzuti, M., 2013. Optimization Methods. Springer Berlin Heidelberg, Berlin, Heidelberg. URL: http://link.springer.com/10.1007/978-3-642-31187-1.

Chang, Y., Lorentzen, R.J., Nævdal, G., Feng, T., 2019. OLYMPUS optimization under geological uncertainty. Computational Geosciences URL: https://doi.org/10.1007/s10596-019-09892-x.

Chaturvedi, A., 2020. Cost-effective Development of Small Fields on Norwegian Continental Shelf. Specialization Project done at NTNU.

Chaturvedi, A., 2021. Scripts for Development of Smart Proxy Models for Well Control Optimization. URL: https://github.com/arpita-ch/TPG4920-Master-Thesis.git.

Chuang, Y.C., Chen, C.T., Hwang, C., 2016. A simple and efficient real-coded genetic algorithm for constrained optimization. Applied Soft Computing 38, 87–105. URL: https://www.sciencedirect.com/science/article/pii/S1568494615006080, doi:10.1016/j.asoc.2015.09.036.

Denney, D., 2010. Pros and Cons of Applying a Proxy Model as a Substitute for Full Reservoir Simulations. Journal of Petroleum Technology 62, 41–42. URL: https://doi.org/10.2118/0710-0041-JPT.

DiCarlo, J., Eustes, A., Steeger, G., 2019. A History of Operations Research Optimization in the Petroleum Industry, in: SPE-196031-MS, Society of Petroleum Engineers, Calgary, Alberta, Canada. p. 20. URL: https://doi.org/10.2118/196031-MS. journal Abbreviation: SPE-196031-MS.

Eberhart, R., Kennedy, J., 1995. A new optimizer using particle swarm theory, in: MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science, pp. 39–43. doi:10.1109/MHS.1995.494215.

Eiben, A.E., Smith, J.E., 2003. Introduction to evolutionary computing. volume 53. Springer. URL: https://link.springer.com/book/10.1007/978-3-662-44874-8.

Fonseca, R.M., Geel, C.R., Leeuwenburgh, O., 2017. Description of OLYMPUS reservoir model for optimization challenge URL: https://www.isapp2.com/optimization-challenge/reservoir-model-description.html.

Fonseca, R.M., Rossa, E.D., Emerick, A.A., Hanea, R.G., Jansen, J.D., 2018. Overview Of The Olympus Field Development Optimization Challenge, European Association of Geoscientists & Engineers. pp. 1–10. URL: https://www.earthdoc.org/content/papers/10.3997/2214-4609.201802246. iSSN: 2214-4609 Issue: 1.

Goodwin, N., 2018. Efficient Large Scale Optimization Under Uncertainty With Multiple Proxy Models URL: https://www.earthdoc.org/content/papers/10.3997/2214-4609.201802302. publisher: European Association of Geoscientists &amp; Engineers.

Grefenstette, J., 2000. Proportional selection and sampling algorithms. Evolutionary Computation , 172–180.

Holland, J.H., 1975. Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, MI. Second edition, 1992.

Hough, P.D., Kolda, T.G., Torczon, V.J., 2001. Asynchronous Parallel Pattern Search for Nonlinear Optimization. SIAM Journal on Scientific Computing 23, 134–156. URL: http://epubs.siam.org/doi/10.1137/S1064827599365823.

Isebor, O.J., Durlofsky, L.J., Echeverría Ciaurri, D., 2014. A derivative-free methodology with local and global search for the constrained joint optimization of well locations and controls. Computational Geosciences 18, 463–482. URL: https://doi.org/10.1007/s10596-013-9383-x, doi:10.1007/s10596-013-9383-x.

Jaber, A.K., Al-Jawad, S.N., Alhuraishawy, A.K., 2019. A review of proxy modeling applications in numerical reservoir simulation. Arabian Journal of Geosciences 12, 701. URL: https://doi.org/10.1007/s12517-019-4891-1.

Jansen, J.D., Durlofsky, L.J., 2017. Use of reduced-order models in well control optimization. Optimization and Engineering 18, 105–132. URL: https://doi.org/10.1007/s11081-016-9313-6, doi:10.1007/s11081-016-9313-6.

Jesmani, M., Bellout, M.C., Hanea, R., Foss, B., 2015. Particle Swarm Optimization Algorithm for Optimum Well Placement Subject to Realistic Field Development Constraints, Society of Petroleum Engineers. URL: https://www.onepetro.org/conference-paper/SPE-175590-MS, doi:10.2118/175590-MS.

Jin, Z.L., Liu, Y., Durlofsky, L.J., 2020. Deep-learning-based surrogate model for reservoir simulation with time-varying well controls. Journal of Petroleum Science and Engineering 192, 107273. URL: https://www.sciencedirect.com/science/article/pii/S0920410520303533, doi:10.1016/j.petrol.2020.107273.

Keras, 2021. Keras documentation: Metrics. URL: https://keras.io/api/metrics/.

Ketkar, N., 2017a. Feed forward neural networks, in: Deep Learning with Python. Berkeley, CA, pp. 17–33. URL: https://doi.org/10.1007/978-1-4842-2766-4_3, doi:10.1007/978-1-4842-2766-4_3.

Ketkar, N., 2017b. Introduction to PyTorch, in: Ketkar, N. (Ed.), Deep Learning with Python: A Hands-on Introduction. Apress, Berkeley, CA, pp. 195–208. URL: https://doi.org/10.1007/978-1-4842-2766-4_12.

Khor, C.S., Elkamel, A., Shah, N., 2017. Optimization methods for petroleum fields development and production systems: a review. Optimization and Engineering 18, 907–941. URL: https://doi.org/10.1007/s11081-017-9365-2.

Kingma, D.P., Ba, J., 2017. Adam: A method for stochastic optimization. URL: https://arxiv.org/abs/1412.6980/.

Kolda, T.G., Lewis, R.M., Torczon, V., 2003. Optimization by Direct Search: New Perspectives on Some Classical and Modern Methods. SIAM Review 45, 385–482. URL: http://epubs.siam.org/doi/10.1137/S003614450242889, doi:10.1137/S003614450242889.

Kolda, T.G., Torczon, V.J., 2003. Understanding Asynchronous Parallel Pattern Search, in: Di Pillo, G., Murli, A. (Eds.), High Performance Algorithms and Software for Nonlinear Optimization. Springer US, Boston, MA. Applied Optimization, pp. 323–342. URL: https://doi.org/10.1007/978-1-4613-0241-4_15.

Louppe, G., Kumar, M., Nahrstaedt, H., 2016. Bayesian optimization with skopt — scikit-optimize 0.8.1 documentation. URL: https://scikit-optimize.github.io/stable/auto_examples/bayesian-optimization.html#problem-statement.

Ma, H., Yu, G., She, Y., Gu, Y., 2019. Waterflooding Optimization under Geological Uncertainties by Using Deep Reinforcement Learning Algorithms URL: https://onepetro.org/SPEATCE/proceedings/19ATCE/3-19ATCE/D031S043R001/217789, doi:10.2118/196190-MS.

Maschio, C., Nakajima, L., Schiozer, D.J., 2008. Production Strategy Optimization Using Genetic Algorithm and Quality Map, Society of Petroleum Engineers. URL: https://www.onepetro.org/conference-paper/SPE-113483-MS, doi:10.2118/113483-MS.

Mohaghegh, S., Hafez, H., Gaskari, R., Haajizadeh, M., Kenawy, M., 2006. Uncertainty analysis of a giant oil field in the middle east using surrogate reservoir model doi:10.2118/101474-MS.

Mohaghegh, S., Modavi, A., Hafez, H., Haajizadeh, M., Guruswamy, S., 2009. Development of Surrogate Reservoir Models (SRM) For Fast Track Analysis of Complex Reservoirs. International Journal of Oil, Gas and Coal Technology 2. doi:10.1504/IJOGCT.2009.023627.

Mohaghegh, S.D., 2007. Smart Completions, Smart Wells and Now Smart Fields; Challenges & Potential Solutions , 67.

Mohaghegh, S.D., 2017. Shale Analytics, in: Mohaghegh, S.D. (Ed.), Shale Analytics: Data-Driven Analytics in Unconventional Resources. Springer International Publishing, Cham, pp. 29–81. URL: https://doi.org/10.1007/978-3-319-48753-3_3, doi:10.1007/978-3-319-48753-3_3.

Nait Amar, M., Zeraibi, N., Jahanbani, A., 2020. Applying hybrid support vector regression and genetic algorithm to water alternating CO2 gas EOR. Greenhouse Gases: Science and Technology 10. doi:10.1002/ghg.1982.

Nait Amar, M., Zeraibi, N., Redouane, K., 2018. Optimization of WAG Process Using Dynamic Proxy, Genetic Algorithm and Ant Colony Optimization. Arabian Journal for Science and Engineering 43, 6399–6412. URL: https://doi.org/10.1007/s13369-018-3173-7, doi:10.1007/s13369-018-3173-7.

Nasir, Y., 2020. Deep reinforcement learning for field development optimization. URL: https://www.researchgate.net/publication/343986479_Deep_Reinforcement_Learning_for_Field_Development_Optimization, arXiv:2008.12627.

Nasir, Y., Volkov, O., Durlofsky, L.J., 2021. A two-stage optimization strategy for large-scale oil field development. Optimization and Engineering URL: https://doi.org/10.1007/s11081-020-09591-y.

Ng, A., 2021. Stanford Engineering Everywhere | CS229 - Machine Learning | Lecture 1 - The Motivation & Applications of Machine Learning. URL: https://see.stanford.edu/Course/CS229/47.

Nielsen, M., 2019. Neural Networks and Deep Learning | BibSonomy. URL: http://neuralnetworksanddeeplearning.com/.

Nocedal, J., Wright, S.J. (Eds.), 2006. Introduction. Springer New York, New York, NY. pp. 1–9. URL: https://doi.org/10.1007/978-0-387-40065-5_1, doi:10.1007/978-0-387-40065-5_1.

Nwankwor, E., Nagar, A.K., Reid, D.C., 2013. Hybrid differential evolution and particle swarm optimization for optimal well placement. Computational Geosciences 17, 249–268. URL: https://doi.org/10.1007/s10596-012-9328-9.

Peng, L., Zhu, Q., Lv, S.X., Wang, L., 2020. Effective long short-term memory with fruit fly optimization algorithm for time series forecasting. Soft Computing 24, 15059–15079. URL: https://doi.org/10.1007/s00500-020-04855-2.

Ridha, B., Mansoori, G.A., 2005. An introduction to artificial intelligence applications in petroleum exploration and production. Journal of Petroleum Science and Engineering - J PET SCI ENGINEERING 49, 93–96. doi:10.1016/j.petrol.2005.09.001.

Russell, S., Norvig, P., 2003. Artificial Intelligence, A Modern Approach. Second Edition.

Sadigov, S., 2019. Well Placement Optimization using Open-Source Simulators URL: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2634869. accepted: 2020-01-03T15:00:17Z Publisher: NTNU.

Sayyafzadeh, M., Alrashdi, Z., 2019. Well controls and placement optimisation using response-fed and judgement-aided parameterisation: Olympus optimisation challenge. Computational Geosciences URL: https://doi.org/10.1007/s10596-019-09891-y, doi:10.1007/s10596-019-09891-y.

Shahkarami, A., Mohaghegh, S., Gholami, V., Haghighat, S., 2014. Artificial Intelligence (AI) Assisted History Matching. doi:10.2118/169507-MS.

Singh, D., Singh, B., 2020. Investigating the impact of data normalization on classification performance. Applied Soft Computing 97, 105524. URL: https://www.sciencedirect.com/science/article/pii/S1568494619302947, doi:10.1016/j.asoc.2019.105524.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research 15, 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

Tang, L., Li, J., Lu, W., Lian, P., Wang, H., Jiang, H., Wang, F., Jia, H., 2021. Well Control Optimization of Waterflooding Oilfield Based on Deep Neural Network. Geofluids 2021, e8873782. URL: https://www.hindawi.com/journals/geofluids/2021/8873782/, doi:10.1155/2021/8873782.

Teixeira, A.F., Secchi, A.R., 2019. Machine learning models to support reservoir production optimization. IFAC-PapersOnLine 52, 498–501. URL: https://www.sciencedirect.com/science/article/pii/S2405896319301971, doi:10.1016/j.ifacol.2019.06.111.

Twomey, J.M., Smith, A.E., 1995. Performance measures, consistency, and power for artificial neural network models. Mathematical and Computer Modelling 21, 243–258. URL: https://www.sciencedirect.com/science/article/pii/0895717794002075, doi:10.1016/0895-7177(94)00207-5.

Van Doren, J.F.M., Markovinović, R., Jansen, J.D., 2006. Reduced-order optimal control of water flooding using proper orthogonal decomposition. Computational Geosciences 10, 137–158. URL: https://doi.org/10.1007/s10596-005-9014-2.

Volkov, O., Bellout, M., 2018. Gradient-based constrained well placement optimization. 1052-1066 URL: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2607512, doi:10.1016/j.petrol.2018.08.033. accepted: 2019-08-08T08:12:39Z Publisher: Elsevier.

Wang, X., Haynes, R.D., He, Y., Feng, Q., 2019. Well control optimization using derivative-free algorithms and a multiscale approach. Computers & Chemical Engineering 123, 12–33. URL: http://www.sciencedirect.com/science/article/pii/S0098135418305088, doi:10.1016/j.compchemeng.2018.12.004.

Ying, X., 2019. An Overview of Overfitting and its Solutions. Journal of Physics: Conference Series 1168, 022022. URL: https://doi.org/10.1088/1742-6596/1168/2/022022, doi:10.1088/1742-6596/1168/2/022022. publisher: IOP Publishing.

Appendix

Github Link

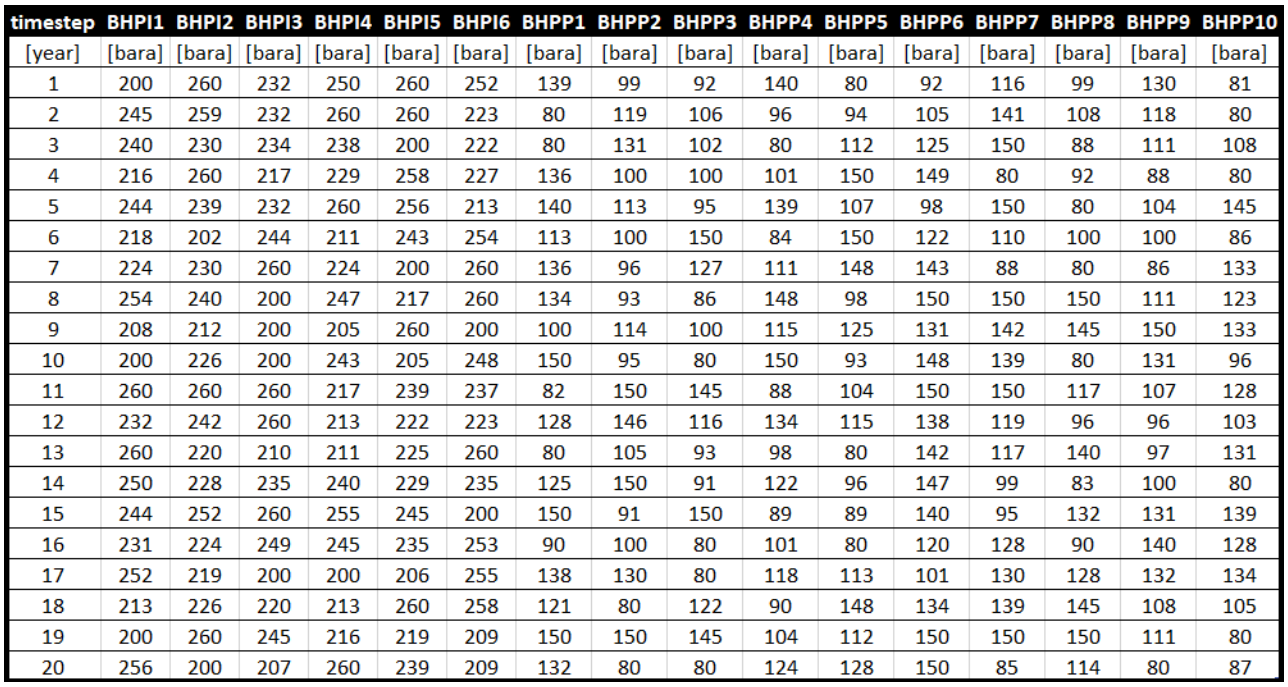
All the python scripts used to achieve objective of this study are uploaded on Github and can be accessed by clicking on the link below.

https://github.com/arpita-ch/TPG4920-Master-Thesis.git

If any issues encountered while accessing and using these scripts, the author can be contacted at arpitachturvedi@gmail.com.

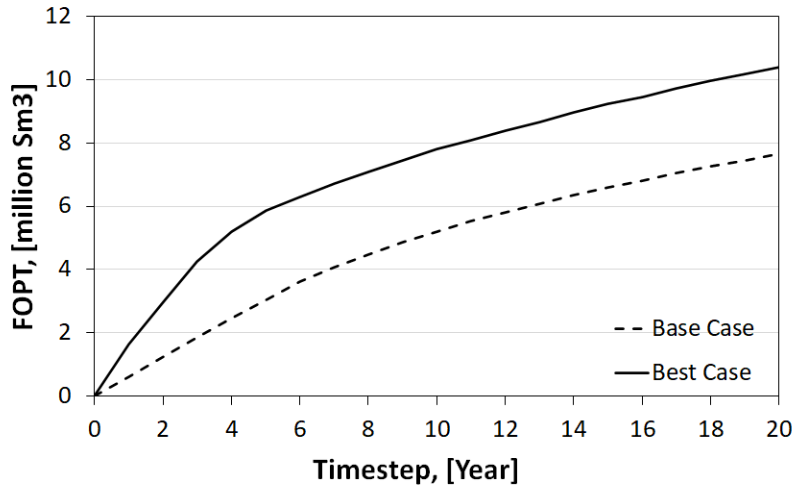
Best Case Results

Table A.1: Optimal BHP configuration from best GA run

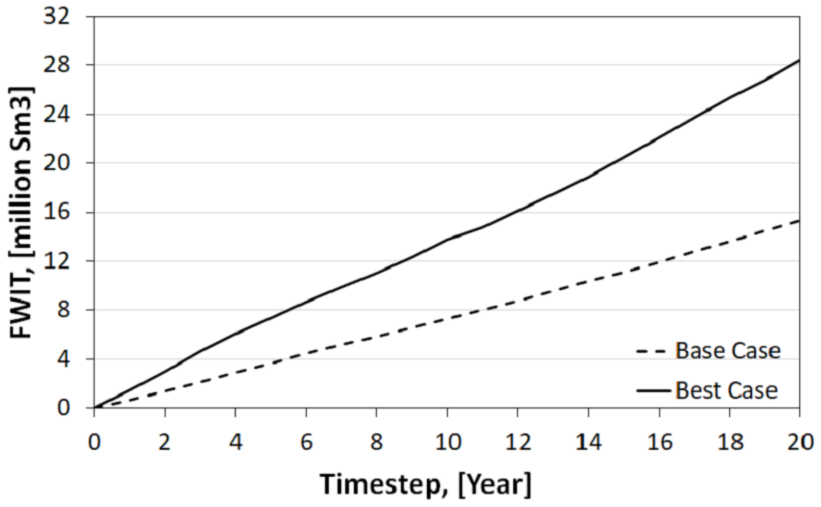


timestep	BHPI1	BHPI2	BHPI3	BHPI4	BHPI5	BHPI6	BHPP1	BHPP2	BHPP3	BHPP4	BHPP5	BHPP6	BHPP7	BHPP8	BHPP9	BHPP10
[year]	[bara]	[bara]	[bara]	[bara]	[bara]	[bara]	[bara]	[bara]	[bara]	[bara]	[bara]	[bara]	[bara]	[bara]	[bara]	[bara]
1	200	260	232	250	260	252	139	99	92	140	80	92	116	99	130	81
2	245	259	232	260	260	223	80	119	106	96	94	105	141	108	118	80
3	240	230	234	238	200	222	80	131	102	80	112	125	150	88	111	108
4	216	260	217	229	258	227	136	100	100	101	150	149	80	92	88	80
5	244	239	232	260	256	213	140	113	95	139	107	98	150	80	104	145
6	218	202	244	211	243	254	113	100	150	84	150	122	110	100	100	86
7	224	230	260	224	200	260	136	96	127	111	148	143	88	80	86	133
8	254	240	200	247	217	260	134	93	86	148	98	150	150	150	111	123
9	208	212	200	205	260	200	100	114	100	115	125	131	142	145	150	133
10	200	226	200	243	205	248	150	95	80	150	93	148	139	80	131	96
11	260	260	260	217	239	237	82	150	145	88	104	150	150	117	107	128
12	232	242	260	213	222	223	128	146	116	134	115	138	119	96	96	103
13	260	220	210	211	225	260	80	105	93	98	80	142	117	140	97	131
14	250	228	235	240	229	235	125	150	91	122	96	147	99	83	100	80
15	244	252	260	255	245	200	150	91	150	89	89	140	95	132	131	139
16	231	224	249	245	235	253	90	100	80	101	80	120	128	90	140	128
17	252	219	200	200	206	255	138	130	80	118	113	101	130	128	132	134
18	213	226	220	213	260	258	121	80	122	90	148	134	139	145	108	105
19	200	260	245	216	219	209	150	150	145	104	112	150	150	150	111	80
20	256	200	207	260	239	209	132	80	80	124	128	150	85	114	80	87

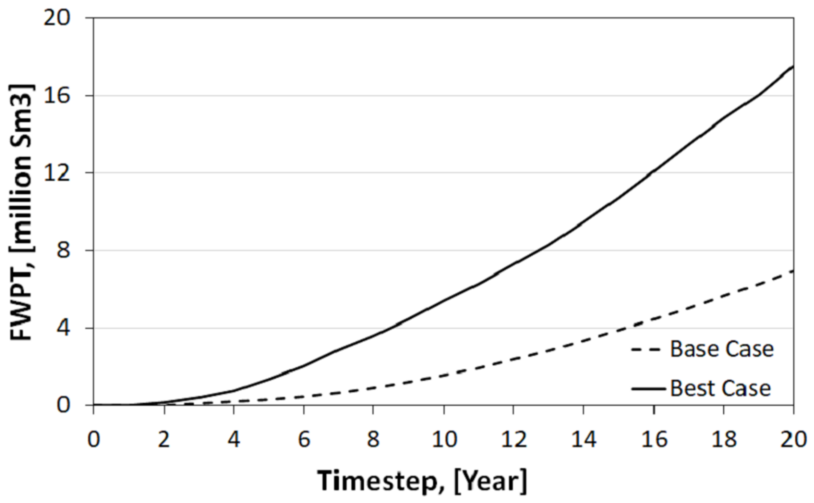
Figure A.2: Comparison of FOPT, FWIT and FWPT for base case and best case.



(a) Cumulative Oil Production

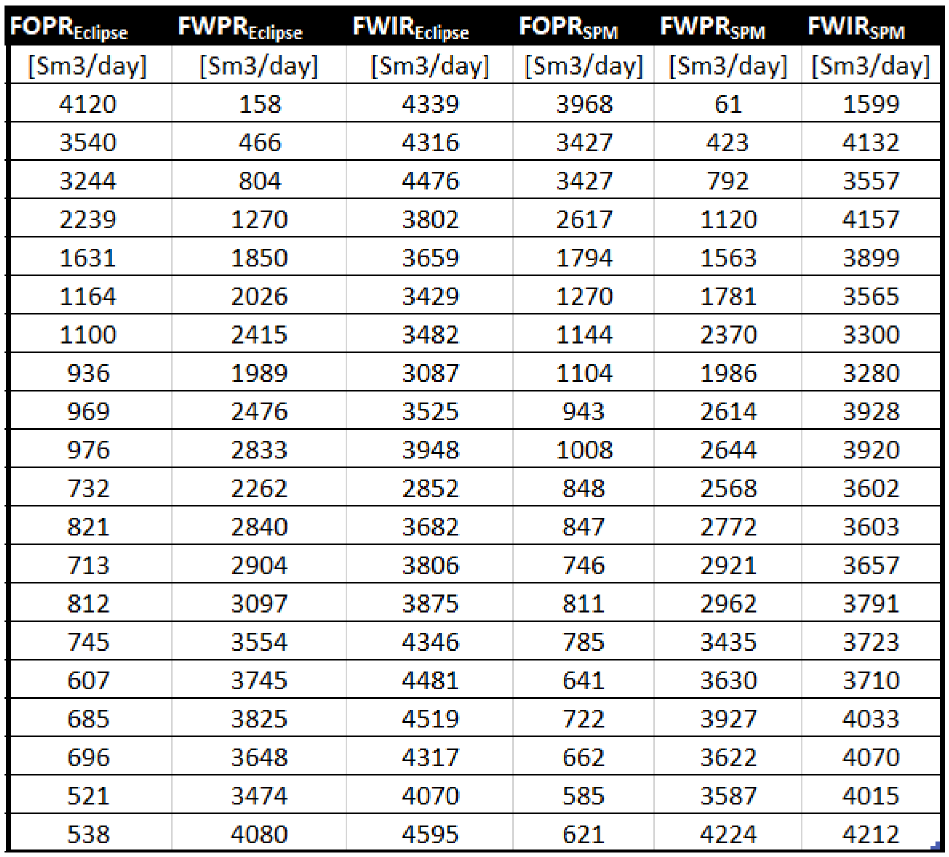


(b) Cumulative Water Injection



(c) Cumulative Water Production

Table A.3: Proxy emulated outputs and Eclipse simulator outputs for optimal BHP setting



$\text{FOPR}_{\text{Eclipse}}$	$\text{FWPR}_{\text{Eclipse}}$	$\text{FWIR}_{\text{Eclipse}}$	FOPR_{SPM}	FWPR_{SPM}	FWIR_{SPM}
[Sm3/day]	[Sm3/day]	[Sm3/day]	[Sm3/day]	[Sm3/day]	[Sm3/day]
4120	158	4339	3968	61	1599
3540	466	4316	3427	423	4132
3244	804	4476	3427	792	3557
2239	1270	3802	2617	1120	4157
1631	1850	3659	1794	1563	3899
1164	2026	3429	1270	1781	3565
1100	2415	3482	1144	2370	3300
936	1989	3087	1104	1986	3280
969	2476	3525	943	2614	3928
976	2833	3948	1008	2644	3920
732	2262	2852	848	2568	3602
821	2840	3682	847	2772	3603
713	2904	3806	746	2921	3657
812	3097	3875	811	2962	3791
745	3554	4346	785	3435	3723
607	3745	4481	641	3630	3710
685	3825	4519	722	3927	4033
696	3648	4317	662	3622	4070
521	3474	4070	585	3587	4015
538	4080	4595	621	4224	4212