Natural Resources Research, Vol. 30, No. 3, June 2021 (© 2021)
https://doi.org/10.1007/s11053-021-09844-2

Original Paper

Smart Proxy Modeling of a Fractured Reservoir Model for Production Optimization: Implementation of Metaheuristic Algorithm and Probabilistic Application

Cuthbert Shang Wui Ng,[1,3] Ashkan Jahanbani Ghahfarokhi,[1] Menad Nait Amar,[2] and Ole Torsæter[1]

Received 23 August 2020; accepted 13 February 2021
Published online: 8 March 2021

Numerical reservoir simulation has been recognized as one of the most frequently used aids in reservoir management. Despite having high calculability performance, it presents an acute shortcoming, namely the long computational time induced by the complexities of reservoir models. This situation applies aptly in the modeling of fractured reservoirs because these reservoirs are strongly heterogeneous. Therefore, the domains of artificial intelligence and machine learning (ML) were used to alleviate this computational challenge by creating a new class of reservoir modeling, namely smart proxy modeling (SPM). SPM is a ML approach that requires a spatio-temporal database extracted from the numerical simulation to be built. In this study, we demonstrate the procedures of SPM based on a synthetic fractured reservoir model, which is a representation of dual-porosity dual-permeability model. The applied ML technique for SPM is artificial neural network. We then present the application of the smart proxies in production optimization to illustrate its practicality. Apart from applying the backpropagation algorithms, we implemented particle swarm optimization (PSO), which is one of the metaheuristic algorithms, to build the SPM. We also propose an additional procedure in SPM by integrating the probabilistic application to examine the overall performance of the smart proxies. In this work, we inferred that the PSO had a higher chance to improve the reliability of smart proxies with excellent training results and predictive performance compared with the considered backpropagation approaches.

KEY WORDS: Reservoir simulation, Dual-porosity dual-permeability, Smart proxy modeling, Backpropagation algorithms, Particle swarm optimization.

INTRODUCTION

Hydrocarbons are among the primary sources of energy in today's world. This is proven by a statistical review conducted by British Petroleum (2020), which found that, in 2019, oil contributed to the largest share of the world primary energy of about 33.1%, whereas natural gas had the third largest share of 24.2%. Hence, they play a pivotal role in quenching the high demand of world energy consumption and such demand will be likely in an upward trend due to the increasing global population (Gerald et al. 2014; International Energy Agency 2018). In addition, the importance of hydrocarbons is reflected by the significant influence of their price on many other major economic do-

[1]Department of Geoscience and Petroleum, Norwegian University of Science and Technology, Trondheim, Norway.
[2]Département Etudes Thermodynamiques, Division Laboratoires, Sonatrach, Boumerdes, Algeria.
[3]To whom correspondence should be addressed; e-mail: cuthbert.s.w.ng@ntnu.no

1520-7439/21/0600-2431/0 © 2021 The Author(s)

mains (Lescaroux and Mignon 2009). This is illustrated clearly by the phenomenon of how many other industries have been affected by the fluctuation of oil price (Lescaroux and Mignon 2009). Therefore, it is essential to have a sustainable hydrocarbon production not only to fulfill the demand for energy consumption, but also to maintain the global economic growth. With respect to this, carbonate reservoirs are one of the main sources of hydrocarbons. These reservoirs make up approximately 60% of the global oil reserves and about 40% of the global gas reserves (Schlumberger 2020b). Most of these reservoirs are naturally fractured, and hence, accurate modeling of fluid flow in these reservoirs is one of the most critical steps to ensure the sustainable production of hydrocarbons.

In general, modeling of fluid flow in porous media can be perceived as a numerical reservoir simulation. Reservoir simulation is one of the most frequently used tools in reservoir management, which is the application of technological, labor, and financial resources to maximize the economic performance and the hydrocarbon recovery of a reservoir (Wiggins and Startzman 1990). This is because it has been implemented extensively to help predict the performance of a reservoir as well as to provide useful information for uncertainty analysis or any optimization task that includes enhanced oil recovery, hydraulic fracturing, and so forth. However, one of the challenges of accurate modeling of fractured reservoirs stems from a lack of underlying theory or principle to describe the behavior of fluid flow in these reservoirs. To mitigate this challenge, Barenblatt (1960) established a theory pertaining to fluid flow in fractured porous media. Based on this theory, Warren and Root (1963) developed the dual-porosity method, which has been one of the most fundamental tools in simulating a fractured reservoir. However, this conventional model does not sufficiently capture the realistic behavior of fluid flow as fluid is assumed to move only through fractures, whereas the matrix blocks only supply fluid to fractures. Hence, this model was enhanced to the dual-porosity dual-permeability (DPDP) model, in which the transport of fluid between matrix blocks is considered (Uleberg and Kleppe 1996). The details regarding this model are explained further below.

Having developed the DPDP model implies that fractured reservoirs can be simulated numerically. Nonetheless, another challenge in terms of computational effort arises as the complexity of simulated fractured reservoirs increases (including as much details as possible to ''describe realistically'' a reservoir). Therefore, reservoir management might not be sufficiently efficient to keep up with sustainable hydrocarbon production. Fortunately, in today's world of digitalization, methods of artificial intelligence and machine learning (AI&ML) have come to the rescue. In this context, Ertekin and Sun (2019) provided a very comprehensive review on the implementation of AI&ML methods in the field of reservoir engineering. They also proposed the use of hand-shaking protocol that would combine the advantages of both traditional and intelligent reservoir modeling to develop more powerful computational protocols. With this, the great potential and extensive utilization of AI&ML-based methods have also been demonstrated further in many technical domains of the petroleum industry (Mohaghegh 2000a, b, c; Parada and Ertekin 2012; Nait Amar and Jahanbani Ghahfarokhi 2020; Nait Amar et al. 2020). Moreover, with the help of AI&ML, Mohaghegh (2011) has coined a new class of reservoir modeling, namely smart proxy modeling (SPM). Fundamentally, SPM is the development of an artificial neural network (ANN) that receives both input and output data from a reservoir simulation model and undergoes a training phase. After the ANN has been trained to recognize the pattern induced by the data (relationship between input and output), it can yield the estimated result that matches with that produced by the reservoir model within a few seconds or minutes. Therefore, this ANN is termed ''smart proxy.'' Regarding this, the word ''smart'' reveals the ability to learn and capture the underlying physical behavior of a simulated reservoir model through pattern recognition and the word ''proxy'' denotes to act on behalf of the original model (Mohaghegh 2017, 2018).

For the past decade, SPM has been considered as a technological breakthrough in the petroleum industry as it has not only reduced the reservoir simulation time significantly, but it also provided the results within an acceptable range of accuracy. The successful application of smart proxies has been demonstrated in many literatures of the oil and gas industry. Mohaghegh et al. (2006) developed surrogate reservoir model (the initial nomenclature of SPM), which was an accurate representation of a sophisticated full-field reservoir model, and used it for uncertainty analysis. With this breakthrough, these surrogate models were implemented on different real fields in Saudi Arabia for geological uncertainty analysis (Mohaghegh et al. 2012a, c). Mohaghegh et al. (2012b, 2015) then reformulated the concept of SPM by categorizing it as grid-based and well-based. As the nomenclatures imply, grid-based SPM is done for the analysis of numerical model at grid block level, whereas well-based SPM is for the analysis at well level. Grid-based SPM has been applied in several real-life CO_2 sequestration projects (Mohaghegh et al. 2012b), whereas well-based SPM has been implemented for optimization of production scheduling of a real field in United Arab Emirates (Mohaghegh et al. 2015). Besides, the application of SPM was then extended gradually to other domains, such as history matching and enhanced oil recovery (EOR). He et al. (2016) coupled the use of SPM with differential evolution (DE) to perform automatic history matching. Alenezi and Mohaghegh (2016) also built a SPM that reproduced and forecasted the dynamic properties of a reservoir that has been water-flooded. Moreover, Mohaghegh (2018) discussed the utilization of SPM under the context of CO_2-EOR as a storage mechanism. Furthermore, Parada and Ertekin (2012) applied SPM to establish successfully a new screening tool for four different improved oil recovery (IOR) methods, including waterflooding, miscible injection of CO_2 and N_2, and injection of steam. Therefore, these literatures do not only show the high applicability of SPM in oil and gas industry, but they also highlight its potential for further enhancement.

Nevertheless, SPM still has few disadvantages. One of them is that a smart proxy built can only be applied to predict what the simulated reservoir might estimate only if the physics assumed in the numerical simulation is not changed. For instance, if a smart proxy is developed on a reservoir model with reservoir pressure of 4000 psia,[1] then it cannot be applied to perform any estimation of parameters when the reservoir pressure is not 4000 psia. To handle this problem, another smart proxy needs to be established. In addition to this, the spatio-temporal database is considered as the backbone of the SPM as it is the main component used to train the ANN model. Thus, if another smart proxy is built (as previously mentioned), then the database needs to be prepared again. Despite having such inconvenience, the time spent on preparation of this database is still much less than the time spent by numerical simulation. Pertaining to this, the preparation of a spatio-temporal database might take about few hours (or for few minutes with the help of commercial software). However, for a sophisticated reservoir simulation model, the computation might take a few days. It is important to understand that smart proxy is another example of data-driven model as it is developed by analyzing the collected data (Alenezi and Mohaghegh 2016, 2017). Hence, careful attention is required when a spatio-temporal database is created. If incorrect data are provided to the smart proxy, it will ‘‘learn wrongly’’ and produce unsatisfactory results. This complies with the short phrase that goes ‘‘garbage in and garbage out.’’

Although there are many literatures explaining the theoretical basis of SPM, it is still treated as ‘‘black-box’’ as commercial software is mostly used to build a smart proxy. Thus, in this work, one of the objectives was to provide a more vivid illustration of how SPM can be performed based on a synthetic reservoir model. Besides, we present another alternative of training algorithm apart from the backpropagation algorithm that is mostly used in SPM. More intriguingly, we include a probabilistic application to evaluate further the overall performance of the developed SPMs. We opine that this integration in SPM is insightful as it helps to better reflect the performance of the proxy models. After this introduction, we discuss briefly the mathematical concepts of the DPDP model and how ANN operates. Three different algorithms, which are two examples of backpropagation algorithms, namely stochastic gradient descent (SGD) and adaptive moment estimation (Adam) algorithms, and particle swarm optimization (PSO), were implemented as the learning algorithm to train the ANN. Hence, the fundamentals of these algorithms are discussed next. Then, we explicate the background of the reservoir model simulated based on the DPDP method and the problem setting of the production optimization case. We also explain how the respective SPM is developed upon it and used in production optimization. Then, the results and discussion will follow. Prior to proceeding to conclusions, we also provide another case study, which considers a heterogeneous fractured reservoir model, to further show the robustness of the methodology discussed in this paper.

[1] 1 psia = 6894.75728 Pa.

METHODOLOGY

Fundamentals of DPDP Model

In the conventional dual-porosity model, a grid block consists of two portions—the matrix block and the fractures. In this model, the fluid flows mainly through the fractures, whereas the matrix blocks only provide fluids to the fracture (Uleberg and Kleppe 1996). This phenomenon of fluid flow is illustrated in a two-dimensional case as in Figure 1.

Assuming a one-dimensional and one-phase flow case, the transport of fluid through the fracture can be mathematically expressed as (Barrenblatt 1960; Warren and Root 1963):

$$\frac{\partial}{\partial x}\left(\frac{k}{\mu B}\frac{\partial P}{\partial x}\right)_{\text{fracture}} + \hat{q}_{\text{matrix\_fracture}} = \frac{\partial}{\partial t}\left(\frac{\emptyset}{B}\right)_{\text{fracture}} \tag{1}$$

where k is permeability, B is the formation volume factor, μ is viscosity of fluid, and $\varnothing$ is porosity. The term $\hat{q}_{\text{matrix\_fracture}}$ indicates the supply of fluid to fractures by the matrix block, and its mathematical expression is:

$$-\hat{q}_{\text{matrix\_fracture}} = \frac{\partial}{\partial t}\left(\frac{\emptyset}{B}\right)_{\text{matrix}} \tag{2}$$

Because the assumption of no fluid flow between the blocks of matrix is not realistic, the dual-porosity model was extended to the DPDP model by adding a flow term in Eq. (2) (Uleberg and Kleppe 1996). Hence, the system of equations representing the DPDP model is:

$$\frac{\partial}{\partial x}\left(\frac{k}{\mu B}\frac{\partial P}{\partial x}\right)_{\text{fracture}} + \hat{q}_{\text{matrix\_fracture}} = \frac{\partial}{\partial t}\left(\frac{\emptyset}{B}\right)_{\text{fracture}} \tag{3}$$

$$\frac{\partial}{\partial x}\left(\frac{k}{\mu B}\frac{\partial P}{\partial x}\right)_{\text{matrix}} - \hat{q}_{\text{matrix\_fracture}} = \frac{\partial}{\partial t}\left(\frac{\emptyset}{B}\right)_{\text{matrix}} \tag{4}$$

Regarding the exchange term, it can be further represented as:

$$-\hat{q}_{\text{matrix\_fracture}} = \sigma\frac{k_{\text{matrix}}}{\mu}\left(P_{\text{matrix}} - P_{\text{fracture}}\right) \tag{5}$$

where P denotes pressure, whereas σ is the shape factor or the geometric factor. This shape factor represents the geometry of the matrix block, and it dictates the flow fluid between the matrix blocks and the fracture system (Kazemi et al. 1976). There are many mathematical formulations available in the literature to describe this shape factor depending upon the physical effects and mechanisms considered (Warren and Root 1963; Ahmad and Olivier 2008; Su et al. 2013). In this context, one of the most widely applied forms is the one proposed by Kazemi et al. (1976), and it was used in this study. Regarding its formulation, Kazemi et al. (1976) discussed that the shape factor can be computed in a three-dimensional case as:

$$\sigma = 4 \times \left[\frac{1}{L_x^2} + \frac{1}{L_y^2} + \frac{1}{L_z^2}\right] \tag{6}$$

where the L term refers to the dimension of the matrix block in x-, y-, and z- directions.

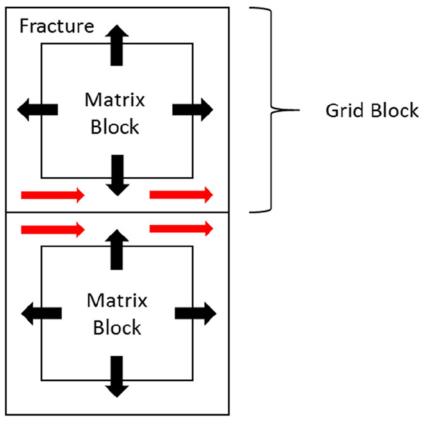


Figure 1. Fluid flow behavior in a dual-porosity model for two-dimensional case. The red arrows indicate the flow thorough fracture network, whereas the black arrows denote the supply of fluid from matrix.

ANN

ANN is a biologically inspired mathematical model or algorithm that can predict any relevant output within an acceptable range of accuracy after learning the relationship between the inputs and outputs provided (Wilamowski and Irwin 2011; Buduma and Locasio 2017). This biological inspiration stems from the imitation of learning method used in human brains. ANN is very robust due to its high generalization ability in capturing the nonlinearity of any process investigated (Gharbi and Mansoori

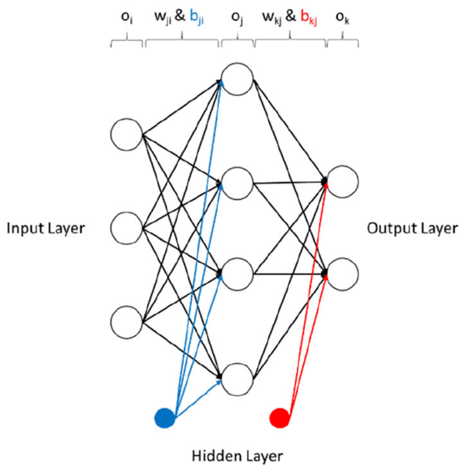


Figure 2. Structure of an ANN.

2005; Wilamowski and Irwin 2011; Nait Amar et al. 2018b). Thus, ANN is better than any traditional regression approach to solve complicated mathematical problems (Gharbi and Mansoori 2005). There are different types of ANN, such as feed-forward neural network, convolutional neural network (CNN), recurrent neural network (RNN). Multilayer perceptron (MLP), which is an example of feed-forward neural network,[2] was implemented here. Regarding the architecture of MLP, it is made up of three different types of layers, namely one input layer, one or more hidden layers, and one output layer (Wilamowski and Irwin 2011; Buduma and Locasio 2017). Each of these layers comprises simple calculating elements, which are known as nodes, units, or artificial neurons (Gharbi and Mansoori 2005). The output from each node in a layer is multiplied by the weights (and biases). The product enters the node in the next layer as input. These inputs are then summed and applied to activation function, also known as transfer function, to produce the output of the node. The structure or topology of an arbitrary ANN that comprises one input layer with three nodes, one hidden layer with four nodes, and one output layer with two nodes is shown in Figure 2.

[2] To avoid confusion, feed-forward neural network, artificial neural network, multilayer perceptron, smart proxy model, smart proxy, and proxy model technically share the same definition in this paper. However, feed-forward neural network is considered as a family of artificial neural network and it includes several types such as multilayer perceptron, radial basis function network, correlation filter neural network.

Referring to Figure 2, the mechanism of ANN can be expounded mathematically as follows. From input layer to hidden layer, the output of the node is computed as:

$$o_j = F\left(\sum_{i=1}^{N_i} w_{ji} o_i + b_{ji}\right) \tag{7}$$

Then, from hidden layer to output layer, the output of the node is calculated as:

$$o_k = F\left(\sum_{j=1}^{N_j} w_{kj} o_j + b_{kj}\right) \tag{8}$$

In Eqs. (7) and (8), the subscript *i* denotes the input layer, the subscript *j* means the hidden layer, and the subscript *k* indicates the output layer, *N* shows the number of nodes in each layer, *o* indicates either the output of node in the current layer or the input of node from previous layer (based upon the subscript), w is a set of weights, and b is a set of biases. Weights are considered as the fitting parameters in modeling of an ANN, whereas bias is an extra node that provides more flexibility for the ANN model to be trained. There are many forms of activation functions *F* that are readily used in ANN modeling. The major ones include sigmoid, rectified linear unit (ReLU), and hyperbolic tangent (Buduma and Locasio 2017). Here, the activation function used was ReLU and it is represented as:

$$F(x) = \begin{cases} 0 \text{for} x \leq 0 \\ x \text{for} x > 0 \end{cases} \tag{9}$$

The derivative of the ReLU function is:

$$\frac{\partial F(x)}{\partial x} = \begin{cases} 0 \text{for} x \leq 0 \\ 1 \text{for} x > 0 \end{cases} \tag{10}$$

Mathematically, ANN learns the relationship or recognizes the pattern between input and output data through the tuning of the sets of weights and biases between the two layers. Through a number of epochs (or iterations), these weights and biases are optimized by minimizing any predefined error function (also known as loss or cost function), such as mean squared error, average absolute percentage error. There are different examples of algorithms that can be used to optimize these weights and biases. Backpropagation algorithm has been commonly used in this context. Examples of backpropagation algorithm are gradient descent (GD), Gauss–Newton algorithm, Levenberg–Marquardt algorithm (LM), adaptive gradient algorithm (AdaGrad), root-mean-square propagation (RMSProp), Adam, and so forth. Additionally, other metaheuristic algorithms, like PSO, DE, genetic algorithm (GA), and so forth, have also been proven useful for neural network training (Nait Amar et al. 2018a, b). As Bianchi et al. (2009) have counseled, metaheuristic algorithm is a high-level mathematical algorithm that is generally natural inspired and used to solve more sophisticated optimization problems. In this study, both backpropagation algorithm and metaheuristic approach have been employed to enable the ANN to "learn." The selected backpropagation algorithm was GD, whereas PSO was the chosen metaheuristic training algorithm.

Backpropagation Algorithm

For the workflow of the GD algorithm, both the inputs and outputs are fed to the ANN as the training phase starts. When the inputs enter the ANN and proceed through the layers, they are gradually processed to yield the predicted output. Thereafter, the predicted output is compared with the desired output. Errors are then propagated back through the ANN. During this backpropagation, the weights and biases are adjusted to minimize the errors. Such process is repeated iteratively until either the errors become less than the predefined tolerance or the number of iterations is reached. The GD is an algorithm that applies the first-order derivative for computation. In this context, the first-order derivative of the error function is implemented to determine the minimum in the error space. The calculation of gradient at iteration t can be expressed mathematically as:

$$g_t = \frac{\partial E(x, w_t)}{\partial w_t} = \left[\frac{\partial E}{\partial w_{1,t}} \frac{\partial E}{\partial w_{2,t}} \frac{\partial E}{\partial w_{3,t}} \cdots \frac{\partial E}{\partial w_{N,t}}\right]^T \qquad (11)$$

where E indicates the error function, $\mathbf{x}$ the input vector, and $\mathbf{w}$ the weight (and bias) vector. Thereafter, the weights are updated by using the following equations. The same idea applies to the updating of the biases.

$$w_{t+1} = w_t + \Delta w_t \qquad (12)$$

$$w_{t+1} = w_t - (\gamma \times g_t) \qquad (13)$$

In Eqs. (12) and (13), the weights (and biases) at iteration $t + 1$ are updated using the weights (and biases) at iteration t, the gradient at t, and γ, which is the learning rate or step size. Therefore, the gradient is always computed at every iteration step to adjust the weights (and biases). Pertaining to the computation of gradient of error function, it is highly dependent on the forms of error function and activation function that were used. Here, the error function used was the mean squared error, whereas the activation function used was ReLU.

The mathematical formulation of the application of GD as learning algorithm is as follows. For the following derivation, the meaning of the subscripts used here is the same as explained above. The term t means the target value or the actual output, P, denotes the total number of training sets provided; thus:

$$\mathrm{E(x, w, b)} = \frac{1}{\mathrm{P}}\sum_{\mathrm{k}=1}^{\mathrm{P}} (\mathrm{t_k} - \mathrm{o_k})^2 \qquad (14)$$

Having defined the error function, the backpropagation algorithm starts by computing the weight update between the hidden and output layers. To perform this computation, the gradient of the error function with respect to the weights between the hidden and output layers is determined. Thereafter, the similar procedure is conducted to calculate the weight update between the hidden and input layer. This algorithm carries on iteratively until the value of error function (obtained by using the updated weights and biases) is less than a predefined tolerance or the initialized number of epochs is reached. For a more substantial understanding of the mathematical formulation of the backpropagation algorithm, peruse Wilamowski and Irwin (2011) and the relevant literatures. Here, the Keras module, which was developed by Chollet (2019), had been implemented with the help of the programming language Python 3.8.1 and TensorFlow 2.1.0 to use the GD algorithm to optimize the weights and biases. However, it is essential to note that in Keras module, instead of using GD algorithm, the stochastic gradient descent (SGD) algorithm is applied. The fundamentals of these two algorithms are the same. The main difference is that, in SGD, the gradient is only computed once at each iteration step (by randomly selecting a sample from the training set) and is used further (Buduma and Locasio 2017). By inducing this stochastic behavior, the computational cost is reduced drastically. Apart from SGD, Adam was another backpropagation algorithm used here; it is a more advanced and robust variant of SGD developed by Kingma and Ba (2015). Mathematically, it approximates the first and second moments of gradients to adaptively calculate the learning rates for different parameters (Kingma and Ba 2015). Refer to Kingma and Ba (2015) for the details of Adam. Here, Adam was also implemented using Python 3.8.1 and TensorFlow 2.1.0.

PSO

PSO was introduced by Kennedy et al. (1995) based upon the simulation of the social behavior of a flock of flying birds. As explained in several literatures (Kennedy et al. 1995; Shi and Eberhart 1999; Nait Amar et al. 2018a), mathematically, this algorithm operates by having a population of particles, which is also known as a swarm of particles. Each of these particles corresponds to a potential position or a solution in a search space. Then, the position of each particle is updated iteratively according to its position and velocity at previous timestep. The movements of the particles in the search space are controlled by their own best-known position (the local best position) and their best-known position in the entire swarm (the global best position). As this process occurs iteratively, the particles in the swarm will eventually converge to an optimal point, which is deemed as the best solution in the search space. The position and velocity for the j^{th} particle in a N-dimensional space at iteration t can be expressed, respectively, as:

$$x_{j,t} = \left\{x_{j1,t}, x_{j2,t}, x_{j3,t}, \ldots, x_{jN,t}\right\} \tag{15}$$

$$v_{j,t} = \left\{v_{j1,t}, v_{j2,t}, v_{j3,t}, \ldots, v_{jN,t}\right\} \tag{16}$$

Then, the velocity of each particle at next iteration $t + 1$ is updated as (Shi and Eberhart 1999):

$$\begin{aligned} v_{jN,t+1} = {} & v_{jN,t} + c_1 r_1\left(pbest_{jN,t} - x_{jN,t}\right) \\ & + c_2 r_2\left(gbest_{N,t} - x_{jN,t}\right) \end{aligned} \tag{17}$$

In Eqs. (15), (16), and (17), $v_{jN,t}$ and $x_{jN,t}$ represent the velocity of the *j*th particle at iteration *t* and its corresponding position in N-dimension quantity, respectively; $pbest_{jN,t}$ corresponds to the N-dimension quantity of the individual *j* at the best position or the local best position at iteration *t*; $gbest_{N,t}$ is the N-dimension quantity of the swarm at the best position or the global best position at iteration *t*; c_1 denotes the cognitive learning factor (also known as cognitive weight), whereas c_2 means the social learning factor (also known as social weight); r_1 and r_2 are random numbers extracted between 0 and 1. Upon updating the velocity, each particle moves to a new potential solution as:

$$x_{jN,t+1} = x_{jN,t} + v_{jN,t+1} \tag{18}$$

A new parameter, inertial weight ω introduced by Shi and Eberhart (1998), was included in Eq. (17) to improve the convergence condition. This also gradually decreases the velocity of the particles to have the swarm of particles under control (Nait Amar et al. 2018a). In other words, it plays a part in balancing the global search also known as exploration, and the local search also termed as exploitation (Shi and Eberhart 1998; Zhang et al. 2015):

$$\begin{aligned} v_{jN,t+1} = {} & \omega v_{jN,t} + c_1 r_1\left(pbest_{jN,t} - x_{jN,t}\right) \\ & + c_2 r_2\left(gbest_{N,t} - x_{jN,t}\right). \end{aligned} \tag{20}$$

In the context of the minimization problem, an objective function f to be minimized is defined. Then, to determine the local best solution at iteration *t* + 1, the following formula is given (Nait Amar et al. 2018a):

$$pbest_{jN,t+1} = \begin{cases} pbest_{jN,t}, \text{iff } \left(pbest_{jN,t}\right) = f(x_{jN,t+1}) \\ x_{jN,t+1}, \text{otherwise} \end{cases} \tag{21}$$

Then, to find the global best solution at iteration *t* + 1, the following mathematical formulation is presented:

$$gbest_{jN,t+1} = \min\left[f\left(pbest_{jN,t+1}\right)\right] \tag{22}$$

In this study, the objective function was the error function in the ANN modeling. To apply PSO as the training algorithm of ANN, this can be simply done by treating the weights and biases as the particles in the algorithm. Hence, the total number of particles in a swarm is the total number of weights and biases. Then, the optimization can be performed using the abovementioned formulations. Here, the package of PySwarms version 1.1.0, which was built by Miranda (2019), was implemented by using the programming language Python 3.8.1 to perform the optimization. In comparison with the SGD algorithm, one of the advantages of PSO is that it is a derivative-free algorithm. This implies that it is more robust as it can be utilized to optimize a mathematical function that is not easily differentiable.

Table 1. Essential parameters used to develop the DPDP reservoir model

Parameters	Values	Units
Initial reservoir pressure	3.47×10^{7}	Pa
Oil density	819.18	kg/m^3
Water density	1041.20	kg/m^3
Oil viscosity	0.0035	Pa s
Water viscosity	0.0005	Pa s

Initial water saturation	Matrix media	Fracture media
Layer 1	0.1922	0.000
Layer 2	0.1924	0.000
Layer 3	0.1926	0.000

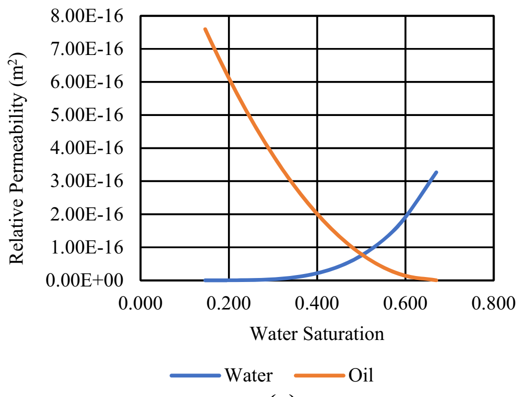
Layer	Matrix block height (m)	Matrix permeability (m^2)	Effective fracture permeability (m^2)	Porosity	
				Matrix media	Fracture media
1	9.144	9.869×10^{-15}	1.480×10^{-12}	0.210	0.0015
2	6.096	1.974×10^{-14}	1.974×10^{-12}	0.230	0.0020
3	12.192	1.480×10^{-14}	2.467×10^{-12}	0.250	0.0018

NUMERICAL SIMULATION MODEL

A three-dimensional, two-phase (black oil and water) reservoir simulation model was built to represent the "true" reservoir model. The "true" reservoir is in fact inspired by the dual-porosity model discussed in Firoozabadi and Thomas (1990), which is a two-dimensional and three-phase model (black oil, water, and gas—including free and dissolved gas). However, most of the reservoir parameters and relevant fluid properties were changed to develop the "true" model. This "true" reservoir model supplied the necessary data for the development of the respective SPM. This reservoir was a DPDP model made up of three layers with uniform thickness.[3] The top of this reservoir was set at the depth of 305 m. About the geometry of this model, each grid block had a length of 25 m, a width of 25 m, and a height of 15.2 m. Thus, the dimension of the reservoir model was 1525 m × 1525 m × 45.7 m, which corresponds to the number of blocks being 61 × 61 × 3. Regarding the well configuration, it was the five-spot pattern in which four injectors were, respectively, set to penetrate near the corners of this reservoir model and a producer was placed in the center of the reservoir. The injectors (producer) would inject water to (would produce from) all the fracture layers. Besides that, the performance of each well in this model was controlled by its respective rate. The target of the field production rate was set equal to the target of the field injection rate for pressure maintenance. For instance, if the target rate of the producer was 400 m^3/day, then the target rate of each of the injector was 100 m^3/day (totaling up to 400 m^3/day of the target of the field injection rate). The numerical simulation of this DPDP reservoir model was conducted using ECLIPSE 100 software Schlumberger (2020a). Other details of this model are summarized in Table 1.

For further clarification, as presented in Table 1, the values of matrix block heights, matrix permeability, and effective fracture permeability were initialized for *x*-, *y*-, and *z*-directions. Additionally, the relative permeability curves and the oil–water capillary pressure curves for matrix media are illustrated in Figure 3. For the two-phase flow in fracture, the linear relationship between relative permeability and saturation, which is also known as "X-curve", is one of the most fundamental models that was determined by Romm (1966). "X-curve" has been used in several fracture-related researches in petroleum industry (Van Golf-Racht 1982; Gilman and Kazemi 1983; Firoozabadi and Thomas 1990). Besides that, regarding the oil–water capillary pressure in the fracture system, it is equal to zero as

[3] In the modeling of DPDP, if three layers are defined, then there will be six resultant layers in which three of them correspond to the matrix system and the remaining three layers correspond to the fracture system. These fluid flow mechanisms of these two systems are represented by extending Eqs. (3), (4), and (5) to three-dimensional and two-phase case.



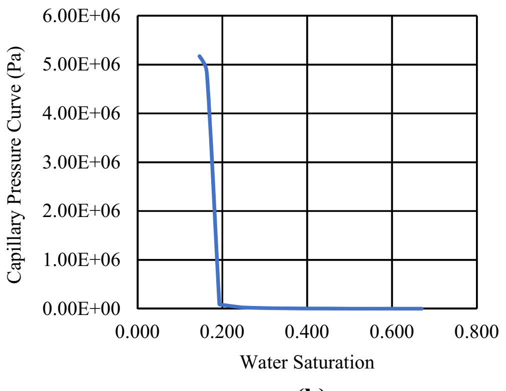


Figure 3. (**a**) Relative permeability curve. (**b**) Oil–water capillary pressure curve for the matrix media.

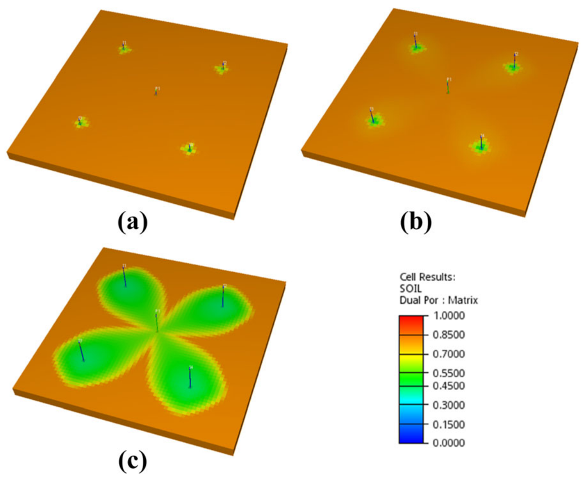


Figure 4. Overview of the matrix system of the reservoir model: (**a**) Layer 1; (**b**) Layer 2; **(c)** Layer 3.

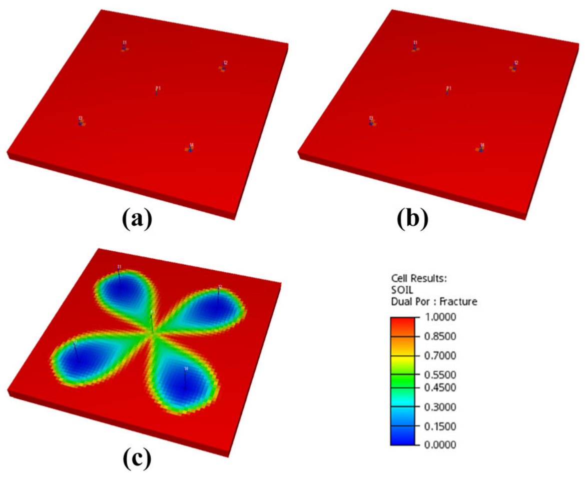


Figure 5. Overview of the fracture system of the reservoir model: (**a**) Layer 1; (**b**) Layer 2; (**c**) Layer 3.

shown in the model discussed by Firoozabadi and Thomas (1990). In short, we selected these models of relative permeability curve and oil–water capillary pressure in both matrix and fracture systems for illustrative purpose. By using the software ResInsight developed by Ceetron Solution AS (2020), this reservoir model depicting oil saturation at the water injection rate of 636 m^3/day (after the injection period of 5 years) is displayed correspondingly in Figure 4 for the matrix system and in Figure 5 for the fracture system.

Based on Figures 4 and 5, much more oil had been swept toward the producers in Layer 3 for both matrix and fracture media. Because the injectors were (the producer was) perforated in all the fracture layers, this denoted that the injected water flowed and swept the oil in (the oil was only produced from) the fracture systems. Given the homogeneity of every layer of the model and the high effective permeabilities in z-direction for all the fracture layers, the cross-flow of fluids between the fracture layers was prominent to contribute to the high sweeping of oil in Layer 3 of the fracture media. This scenario also occurred to the matrix media because it needed to supply the oil to the fracture system where most of the oil has been swept and produced. In this context, we reiterate that the DPDP reservoir modeling was not the main goal of this work. In fact, we intended to design a valid DPDP model to

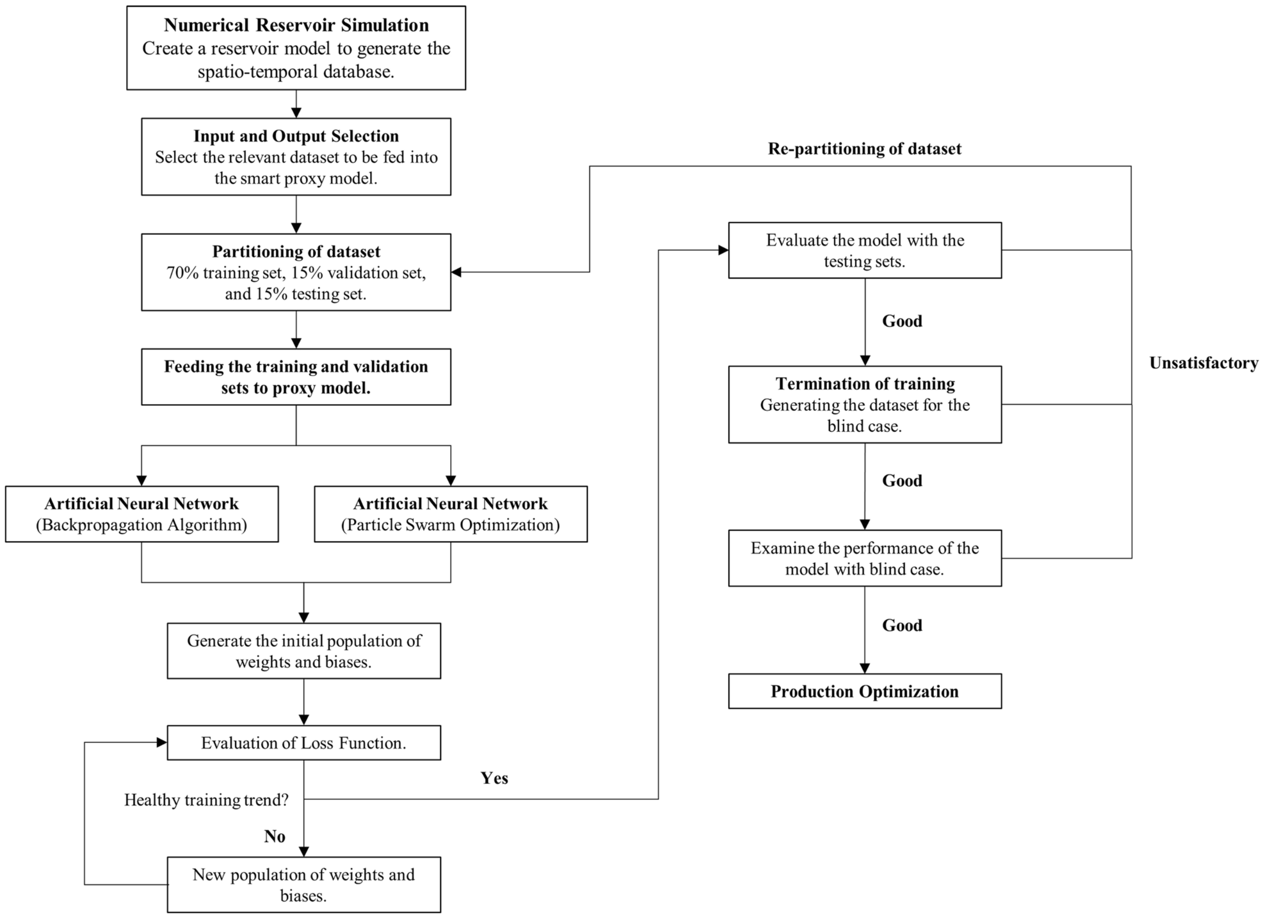

Numerical Reservoir Simulation
Create a reservoir model to generate the spatio-temporal database.
Input and Output Selection
Select the relevant dataset to be fed into the smart proxy model.
Partitioning of dataset
70% training set, 15% validation set, and 15% testing set.
Feeding the training and validation sets to proxy model.
Artificial Neural Network (Backpropagation Algorithm)
Artificial Neural Network (Particle Swarm Optimization)
Generate the initial population of weights and biases.
Evaluation of Loss Function.
Healthy training trend?
Yes
No
New population of weights and biases.
Re-partitioning of dataset
Evaluate the model with the testing sets.
Good
Termination of training
Generating the dataset for the blind case.
Good
Examine the performance of the model with blind case.
Good
Production Optimization
Unsatisfactory

Figure 6. General workflow of SPM.

Table 2. Values of the economic parameters used in this example of production optimization

Parameters	Values	Units
Oil price, P_o	377.40	USD/m^3
Cost of produced water, P_w	44.02	
Cost of injected water, P_{inj}	44.02	
Monthly discounted rate	0.833	%

demonstrate that our developed proxy model was functioning accurately.

PRODUCTION OPTIMIZATION

Smart proxy is widely developed in the petroleum industry because of its inexpensive computational effort. However, SPM is an objective-oriented task, which implies that modelers need to first know what the smart proxy is used for prior to developing it. After identifying the purposes or functions of the model, modelers would have a well-established understanding pertaining to the preparation of the spatio-temporal database (input and output data) used for neural network training. Regarding this, we used an illustrative example of production optimization as the objective of developing the smart proxy. For this illustrative example, we assumed the production lifetime of the reservoir model discussed

Table 3. Simulation scenarios executed for SPM

Scenario Index	Injection rate (m^3/day)
1	636
2	676
3	715
4	755
5	795

Table 4. Selected input and output data

Inputs			Output
Indexes	Simulation scenario	Scenarios 1, 3, and 5	Field oil production rate at time t
Static inputs	Grid block ith position	Well group (grid block kth denotes the perforated kth grid block)	
	Grid block jth position		
	Grid block kth position		
	Porosity	Average values of layers with well perforation, layers of matrix media, layer of fracture media	
	Permeability		
	Matrix block height	Matrix media (parameters in DPDP modeling)	
	Shape factor		
Dynamic inputs	Time	Monthly basis (timestep 0 to timestep 360)	
	Bottom-hole pressure	For 4 injectors and 1 producer at time t at time t at time $t-1$	
	Field water injection rate		
	Field oil production rate		

to be 30 years and the objective function to be the net present value (NPV). In this case, we needed to decide the target of the field injection rate that can maximize the NPV throughout the production lifetime. The NPV for this optimization example can be formulated as:

$$\mathrm{NPV} = \sum\nolimits_{k=0}^{N} \frac{P_o Q_{o,k} - P_w Q_{w,k} - P_{inj} Q_{inj,k}}{(1+r)^k} \tag{23}$$

where the subscripts o, w, and inj denote oil, water (produced), and injected water, respectively; P is the price (or cost) per standard barrel (the corresponding unit is USD/m^3), Q is total amount for a certain timestep (the respective unit is m^3), r is the discount rate, and k is the timestep. To calculate Q, the following equation was used:

$$Q_{i\epsilon\{o,w,inj\},k} = q_{i\epsilon\{o,w,inj\},k} \times \Delta t_k \tag{24}$$

where q is the flow rate reported (either by the numerical simulation or the developed SPM) on monthly basis (the unit is m^3/day) and Δt_k is the number of months for timestep k. Here, the smart proxy for the prediction of injection rates was not developed as the injection rates remained constant throughout the production period of the reservoir model. Hence, for practical purpose, only two SPMs were developed, which, respectively, predicted the oil production rates and the water production rates (both on monthly basis). With respect to this, it is possible to develop a SPM that predicts simultaneously two outputs, namely both oil and water rates. Nevertheless, the tuning of the weights and biases can be more challenging. Thus, for better and more fundamental demonstration of SPM, we decided not to go with this option in this work. Upon formulating the objective function used in this example of production optimization, the setting of the economic parameters[4] used is presented in Table 2.

SMART PROXY MODELING

To build a SPM, the first step is to generate the spatio-temporal database, which is used as the input and output data to train, validate, and test the model. This database is developed by retrieving the essential data from the numerical reservoir simulation. This step is very crucial because the data extracted will determine the usefulness of this proxy model. For this work, the input and output data selected from the ''true'' reservoir model are summarized in Table 4 (the details are explained further below). The database is considered as the backbone of SPM because it is the source of the data used to train the neural network.

Data Preparation and Analysis

To generate data used for the neural network training, five different simulation scenarios, namely the target of the injection rates at 636 m^3/day, 676 m^3/day, 715 m^3/day, 755 m^3/day, and 795 m^3/day, were run (the other parameters used in the numerical reservoir simulation were kept constant). More

[4] We understand that the economic parameters used here might not reflect the real-world case, but our goal here is to present the application of the smart proxy via an illustrative optimization task.

precisely, only three of them were used for the development of smart proxy, whereas the remaining two were used as the blind cases, which are discussed further below. Table 3 summarizes the five simulation scenarios, of which scenarios 1, 3, and 5 were used for SPM.

Upon running the simulations, the spatio-temporal database was readily generated. This database was developed by extracting the static and dynamic data from the numerical simulation. In this context, static data indicate that the data do not change with time (e.g., porosity, permeability), whereas dynamic data denote otherwise (e.g., instance, water injection rate, oil production rate). One of the main challenges of SPM is the humongous size of the spatio-temporal database. This occurs when the geological properties (static properties) of the simulated reservoir model are very heterogeneous (each of the grid blocks in the reservoir model has different values of porosity and permeability). The high geological heterogeneity will cause the SPM to be impractical if all these static data are used. To alleviate this problem, several literatures (Mohaghegh et al. 2012a, b, c, 2015; He et al. 2016; Alenezi and Mohaghegh 2016, 2017) recommend the application of tier system to delineate the reservoir model. In this aspect, the Voronoi graph theory was implemented to re-upscale these static properties through the lumping of the reservoir layers. By doing so, the size of the static inputs used in defining the structure of the spatio-temporal database can be decreased. However, here, despite having a total of 22,326 grid blocks in the reservoir model, it was not considered to be very complex because the porosity and permeability were homogenous per layer. Hence, the reservoir model can be simply delineated by categorizing it into the matrix media and fracture media.

After resolving the issue of reservoir complexity, the selection of input and output data needs to be considered. For a real-life reservoir model, the spatio-temporal database can still be gigantic to be entirely used as the input and output for SPM. To mitigate this challenge, the above-mentioned literatures propose to use the key performance indicator (KPI) coupled with fuzzy logic to help rank the degree of influence of different properties in the selection of input and output, and it is conducted mostly by using commercial software. In this study, for the purpose of illustration, the input and output data for SPM were determined based upon our knowledge of reservoir engineering. Thereafter, the input and output data yielded the final database applied for training, validating, and testing the neural network as summarized in Table 4, which shows 54 static inputs and 8 dynamic inputs.

On the one hand, regarding static properties, the scenario index, which helps the neural network to identify which instance of the injection rates is used, was one of them. Besides this, the well locations make up 25 out of 54 static inputs because there were 5 wells in total and each of the locations was represented as *i*th, *j*th, and *k*th positions of the grid blocks (with all the fracture layers perforated). This corresponded to one group of the static inputs (Table 4). For both porosity and permeability, each of them comprised 11 static inputs, and 5 of them corresponded to the inputs of the average values of grid block where the wells were perforated and the remaining 6 corresponded to the inputs for the 3 layers in both matrix and fracture systems. Thereafter, the heights of the matrix blocks and the shape factors, respectively, contributed to 3 static inputs.

On the other hand, the bottom-hole pressures of all 5 wells contributed to 5 of the 8 dynamic inputs. Besides that, the timestep also acted as one of the dynamic inputs. The water injection rate at time t (on monthly basis) was also a dynamic input. The remaining dynamic input was the oil production rate at time $t-1$ (on monthly basis), whereas the oil production rate at time t (on monthly basis) was used as the output data instead of being treated as input data in this neural network training. For the development of smart proxy for the prediction of water production rates, the input and output data were essentially the same. However, only the oil production rates at time $t-1$ and t needed to be replaced with the water production rates at time t 1 and t. Besides that, each of the simulation scenarios was run for 30 years. Since the oil production rates were reported on monthly basis, this corresponded to 360 months (30 years $\times$ 12 months/year). By starting from timestep = 0, there were a total of 361 timesteps for each scenario. This resulted in a total number of 68,229 records (3 scenarios $\times$ 361 timesteps/scenario $\times$ 63 records/timestep) in the database, which was to be fed into the neural network for training.

Neural Network Training

Training the neural network is the most essential part of SPM. Prior to feeding the input and

Table 5. Ranges of values of training data

Parameters	Minimum value	Maximum value
Time (months)	0	360
Simulation scenario index	1	5
Well location (grid block position)	4	46
Porosity	0.0015	0.2500
Permeability (m^2)	9.869×10^{-15}	2.467×10^{-12}
Matrix block height (m)	6.096	12.192
Shape factor (m)	0.0023	0.0091
Injector bottom-hole pressure (bar[a])	334	355
Producer bottom-hole pressure (bar)	140	345
Field water injection rate (m^3/day)	636	795
Field oil production rate at time t and $t-1$ (m^3/day)	0	795
Field water production rate at time t and $t-1$ (m^3/day)	0	619

[a]1 bar = 100 kPa

Table 6. Topology of the SPM

Type of layers	Number of layers	Number of nodes
Input	1	62
Hidden	1	10
Output	1	1

Table 7. Essential parameters for the SGD and PSO algorithms

SGD and Adam		PSO	
Parameters	Values	parameters	Values
Number of Epochs	2000	Number of Epochs	2000
Step size	0.01	Number of particle swarms	100
		Inertial weight	0.800
		Cognitive weight	1.005
		Social weight	1.050

output data into the ANN for training, the database is normalized between 0 and 1, thus:

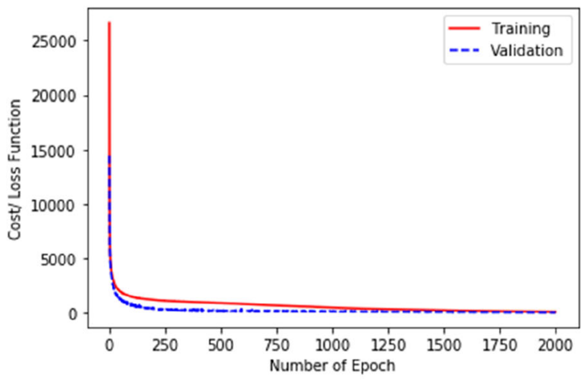
$$x_{normalized} = \frac{x_i - x_{min}}{x_{max} - x_{min}} \tag{25}$$

where x $x_{normalized}$ means the normalized value of x_i, which is the initial data, whereas x_{max} and x_{min}, respectively, indicate the maximum and minimum of data in a group of properties (Table 4). Pertaining to this, the ranges of the values of the training data used are shown in Table 5. By normalizing the data, the convergence condition can be further enhanced, and the ANN is more likely to "learn better" the relationship between the input and output data. Apart from this, the topology of the ANN utilized here is summarized in Table 6. The topology also included two bias nodes, which are not listed in Table 6. One of them was placed in between the input layer and the hidden layer, whereas another one was located between the hidden layer and the output layer.

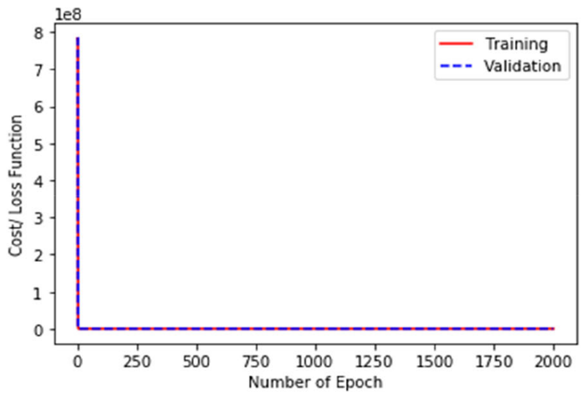
In addition, the relevant parameters required to perform the backpropagation algorithms (SGD and Adam) and PSO algorithms are presented in Table 7. Regarding Adam, there are three other essential parameters, such as exponential decay rates of the estimates of the first and second moments, and constant of numerical stability. Here, the values of these three parameters were, respectively, assigned to be 0.9, 0.99, and 10^{-7}. For PSO, because each of the weight (bias) is treated as one particle, the number of particle swarms indicated the number of sets of particles used in the neural network training.

Thereafter, the normalized database was partitioned into three different sets, which are training, validation, and testing.[5] Here, 70% of the database (47,760 records) was used for training, 15% (10,235 records) for validation, and 15% (10,234 records) for testing. As the training set is fed into the ANN, it enables ANN to capture the underlying physical principles of the simulation by learning the relationship between input and output data. In addition, the validation set ensures that its respective error (loss) reduces, while the error produced by the training set also decreases. This downward trend reflects a healthy behavior of training process. In this study, it was essential to clarify that the validation set did not change the weights and biases (Mohaghegh 2018). It merely uses the weights and biases optimized by the training set to evaluate whether the training process is converging. In other words, the training set was employed to prevent any overtraining or overfitting issue of the ANN (Mohaghegh 2018). Over-fitting occurs if the ANN memorizes the pattern of the data provided and it is unable to give a

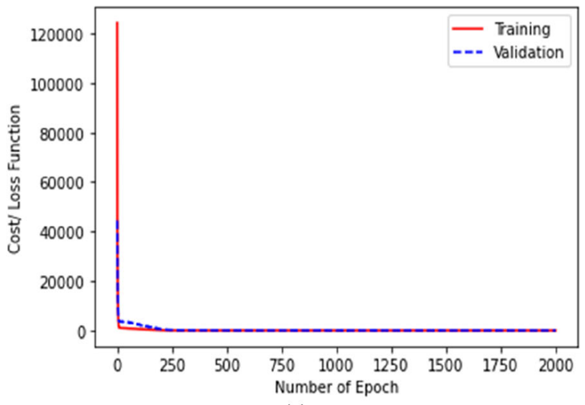
[5] Mohaghegh (2018) discussed that the spatio-temporal database should be divided into three different sets, namely training, calibration, and validation. In this paper, to elude confusion, the calibration set was termed as the validation set, whereas the validation set was referred to as the testing set.



(a)

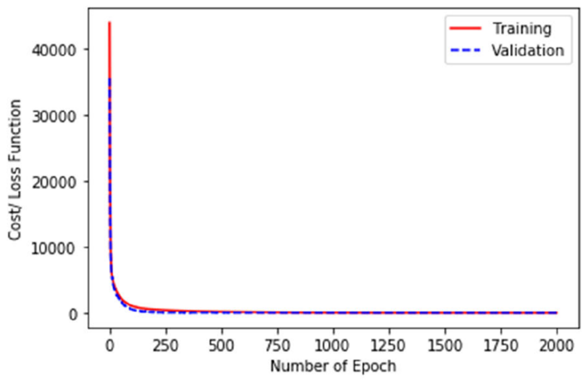


(b)

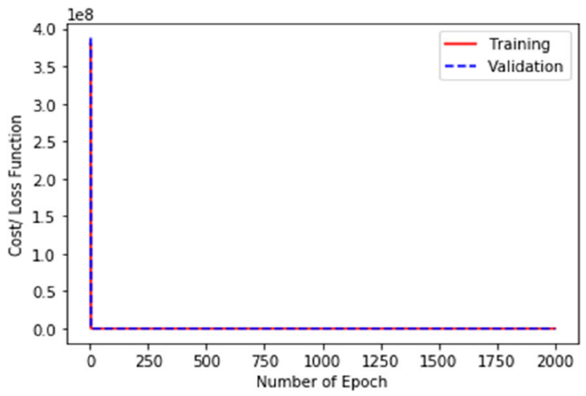


(c)

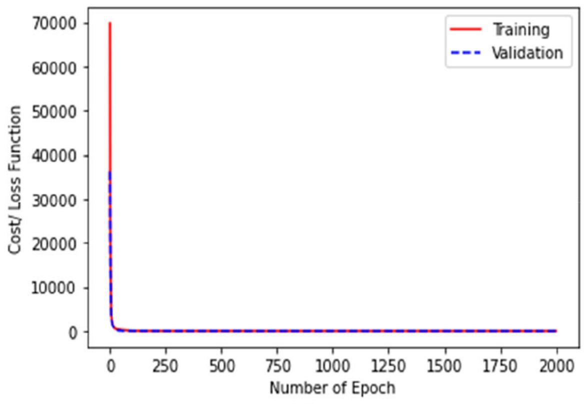
Figure 7. Oil production rate: plots of loss function against number of epochs for the smart proxy trained with (**a**) SGD, (**b**) PSO, and (**c**) Adam.



(a)



(b)



(c)

Figure 8. Water production rate: plots of loss function against number of epochs for the smart proxy trained with (**a**) SGD, (**b**) PSO, and (**c**) Adam.

good prediction when other data are supplied. The testing set assists in checking the predictability of the trained neural network.

After the trained ANN was evaluated by the testing set, it should be provided with a new set of data (that were not from the database) to perform a blind case run. This step is crucial to further confirm the robustness of the developed SPM. Once the results of the training and testing with a blind case run are within acceptable accuracy, the SPM can be employed for further analysis. The general workflow of building a SPM is summarized in Figure 6. As briefly discussed, the error function used in training the ANN was the mean squared error. However, for better evaluation of the performance of the ANN, other metrics including average absolute percentage

Table 8. Performance metrics of the smart proxy for oil rate prediction

		AAPE (%)	RMSE	R^2
Stochastic gradient descent	Training (758 data)	1.770	10.66	0.9954
	Validation (163 data)	1.567	7.512	0.9977
	Testing (162 data)	1.768	7.769	0.9971
Particle swarm optimization	Training (758 data)	0.349	2.378	0.9998
	Validation (163 data)	0.536	14.22	0.9934
	Testing (162 data)	0.352	2.408	0.9998
Adam	Training (758 data)	0.617	1.829	0.9999
	Validation (163 data)	0.649	2.036	0.9998
	Testing (162 data)	0.646	1.487	0.9999

Table 9. Performance metrics of the smart proxy for water rate prediction

		AAPE (%)	RMSE	R^2
Stochastic gradient descent	Training (758 data)	–	1.728	0.9998
	Validation (163 data)	6.461	1.685	0.9998
	Testing (162 data)	8.159	1.652	0.9999
Particle swarm optimization	Training (758 data)	6.565	0.547	0.9999
	Validation (163 data)	–	0.864	0.9999
	Testing (162 data)	7.629	0.761	0.9999
Adam	Training (758 data)	6.753	0.475	0.9999
	Validation (163 data)	4.914	0.262	0.9999
	Testing (162 data)	6.504	0.389	0.9999

error (AAPE%), root-mean-squared error (RMSE), and the correlation coefficient (R^2) were also implemented, and their corresponding formulas are:

$$\mathrm{AAPE}(\%) = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{t_i - o_i}{t_i}\right| \times 100 \tag{26}$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(t_i - o_i)^2} \tag{27}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(t_i - o_i)^2}{\sum_{i=1}^{N}(o_i - -t)^2} \tag{28}$$

where N is total number of data in a set, t_i is the target or actual output value, o_i is the estimated output value, and $-t$ is the mean of the actual output values.

RESULTS AND DISCUSSION

As mentioned above, we built two SPMs to correspondingly predict oil production rates and water production rates at a certain target of injection rate. The topology presented in Table 6 was used to develop these proxy models. For each of these proxy models, the neural network training phase was performed separately by implementing the SGD, PSO, and Adam algorithms. Therefore, precisely speaking, there were 6 SPMs built here. Aside from the neural network training, the validation phase was also done simultaneously to ensure that the trained ANNs have a better generalization capability. Figures 7 and 8 show how the cost function deteriorated as the number of epochs increased in both training and validation phases when SGD, PSO, and Adam were utilized to train the ANN model. This decreasing trend gave a higher confidence that these trained ANN models had good performances in terms of prediction. This decreasing trend further confirmed that these ANNs were prevented from merely memorizing the pattern of the database provided. Thereafter, the testing phase was done to further investigate the predictive performance of the trained neural networks.

The results of the evaluation of the performance of the ANNs are presented in Table 8 for oil production rate prediction and Table 9 for the water production rate prediction. The corresponding cross-

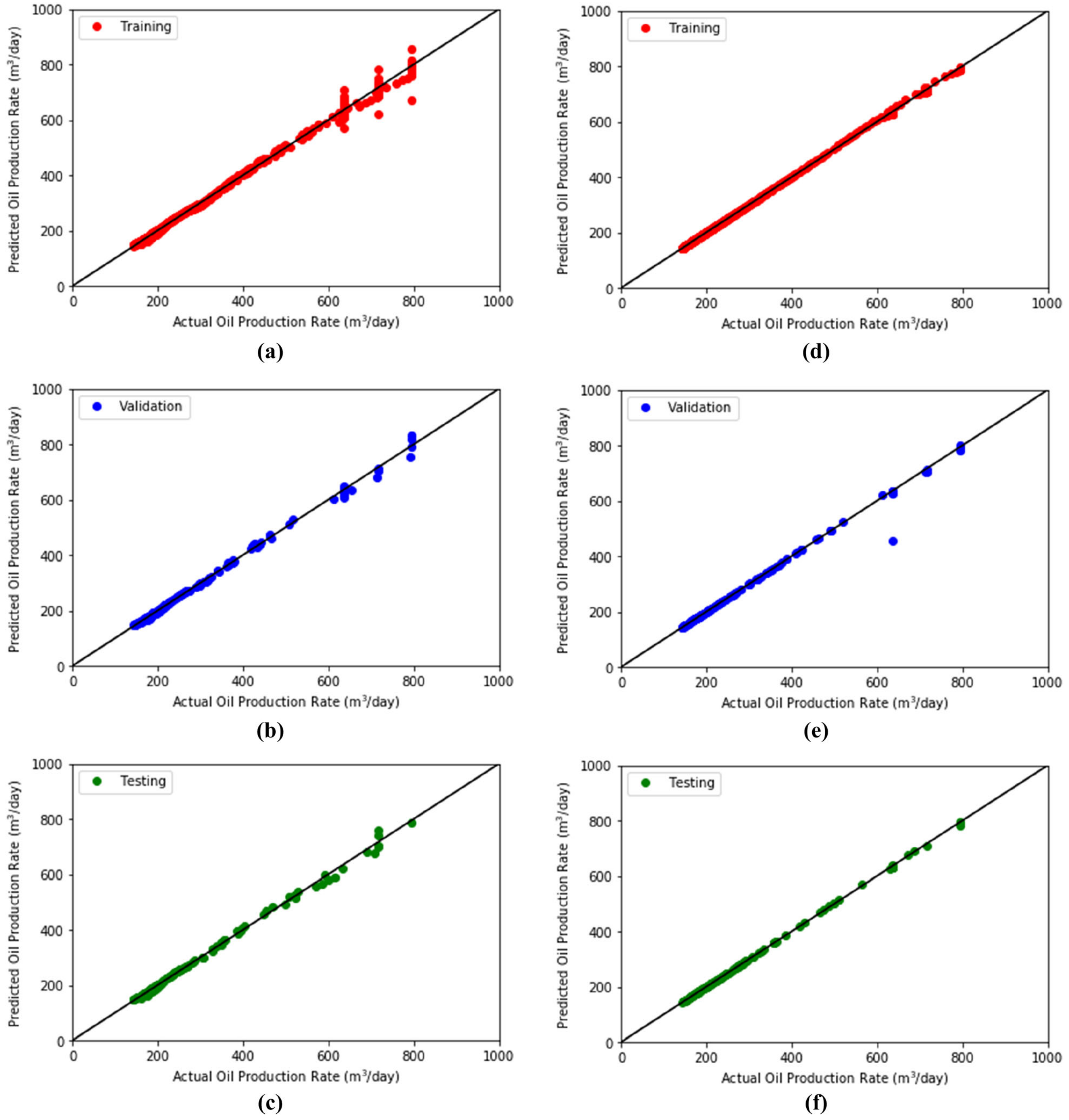
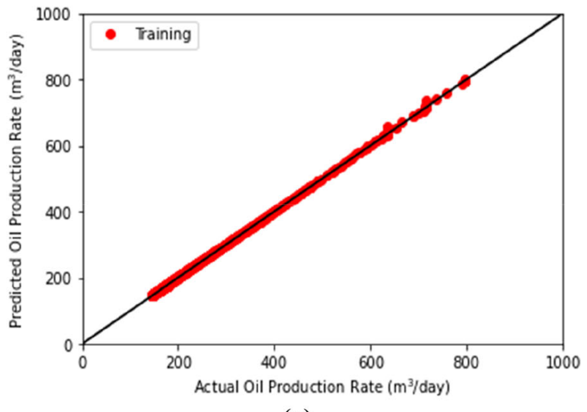
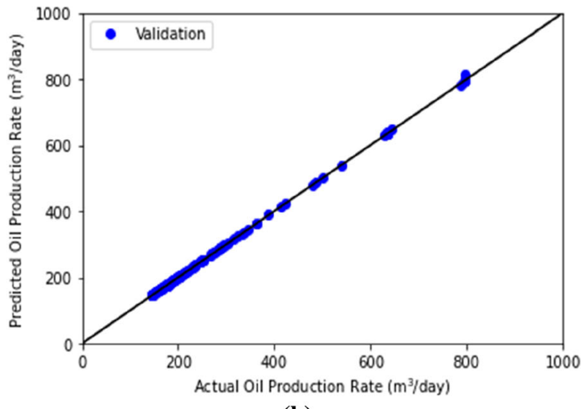


Figure 9. Oil production rate: plots of correlation coefficient (R^2): for SGD **(a)** training, **(b)** validation, **(c)** testing; for PSO **(d)** training, **(e)** validation, **(f)** testing; and for Adam **(g)** training, **(h)** validation, **(i)** testing.

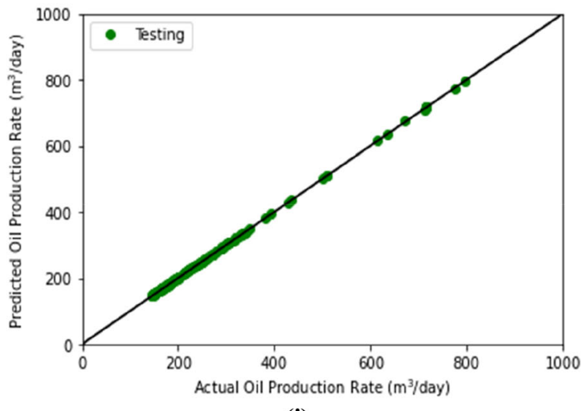
plots between the actual output and the predicted output for the training, validation, and testing sets are illustrated in Figure 9 for oil production rate and Figure 10 for water production rate. Pertaining to the smart proxies for the prediction of oil rate, the results shown in Table 8 indicate that Adam outperformed SGD and PSO in the training, validation, and testing phases in terms of RMSE and correlation coefficient. However, regarding AAPE, PSO had the best performance in all the three phases.



(g)



(h)



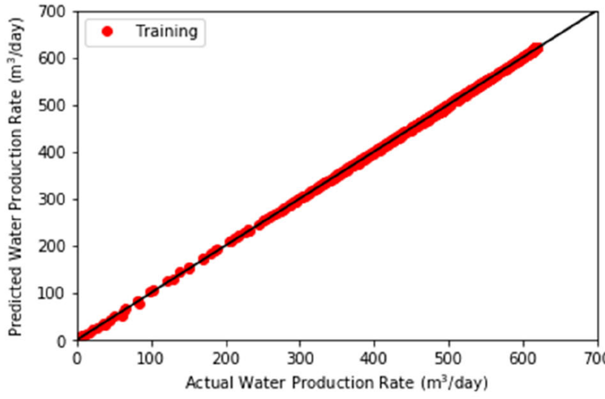
(i)

Figure 9. continued.

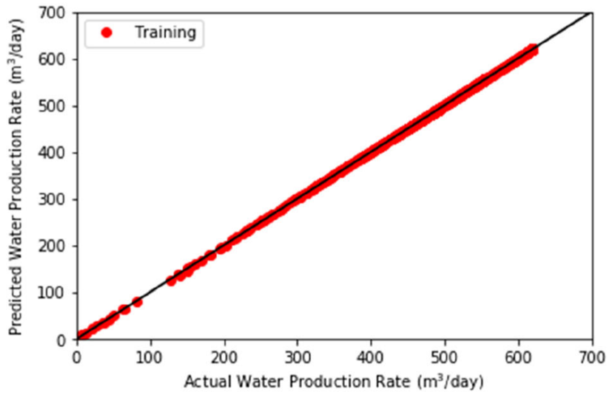
Additionally, better performance of Adam is also presented in Figure 9. As it can be observed, much more data samples lie on the 45-degree line as the Adam was used to develop the smart proxies compared to the cases where the SGD and PSO were utilized. Hence, Adam generally had the best performance, whereas PSO performed better than SGD. Nonetheless, in the validation phase, SGD performed better than the PSO in terms of the minimization of RMSE and the maximization of the correlation coefficient. This can be due to the existence of an over-estimated data point (an outlier) in the validation phase of PSO (as shown in Figure 9e). Because the healthy training process is illustrated in Figure 7, it was deduced that any of these trained models was sufficiently good to be applied to predict the oil production rate. This is further justified by the results of the performance metrics in Table 8, which indicate that the correlation coefficients yielded by all the datasets exceeded 0.99 and both AAPEs and RMSEs exhibited in all the phases were considerably low.

For the prediction of water production rate (as illustrated in Figure 10), it is difficult to infer whether the backpropagation algorithm or the PSO yielded a better performance in the training, validation, and testing phases. However, according to, Adam generally had the best results as compared with SGD and PSO, whereas PSO performed better than SGD. In addition, the results of AAPE were not provided for the training phase of SGD and the validation phase of PSO because, in these phases, there were a few over-estimated data points (outliers) that caused the AAPE to be very large (more than 1000%). This is because when these data points were selected at the early stage of water breakthrough, the actual water production rate was very miniscule. Based on Eq. (26), if the numerator is in the order of magnitude of 1 or 10, then the AAPE will increase drastically. Thus, for practical reasons, the results were not shown here. Despite this, this scenario provided an insight that we needed to look at different performance metrics during SPM to determine whether the built proxy models functioned satisfactorily. Besides, these outliers did not affect the overall predictive capability of the smart proxy built here as the model was still able to capture the general data pattern during the development stage as presented in Figure 10.

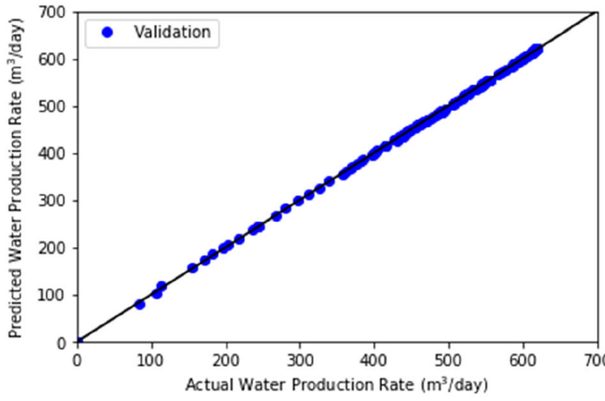
After developing the SPMs, two blind cases were run by using the target of the injection rates at 676 m^3/day and 755 m^3/day to provide more insightful ideas regarding the usefulness of the trained smart proxies. In other words, the spatio-temporal databases when the target of the injection rates was, respectively, at 676 m^3/day and 755 m^3/day created to be fed into the smart proxies to observe how well they can make predictions. It is essential to know that, in order to practically apply



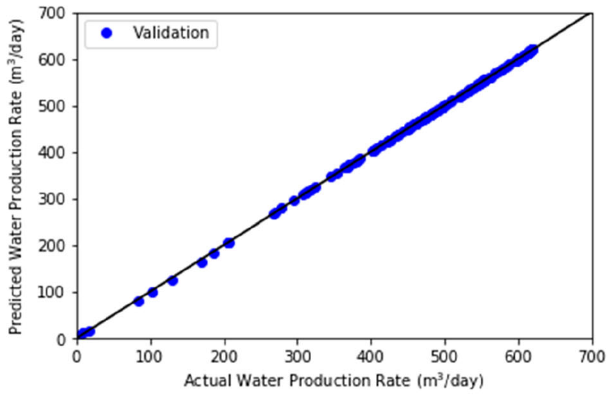
(a)



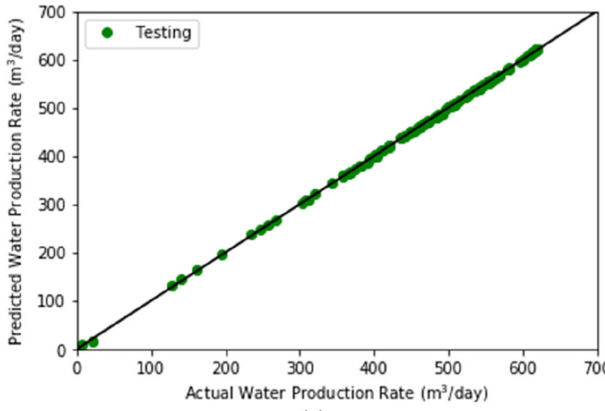
(d)



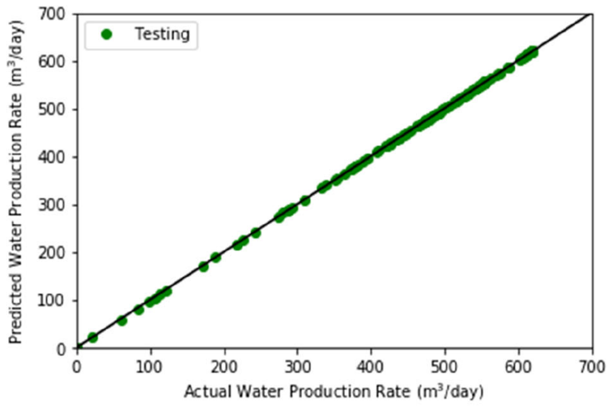
(b)



(e)



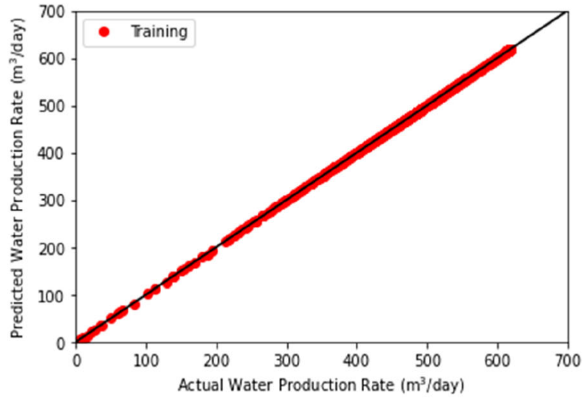
(c)



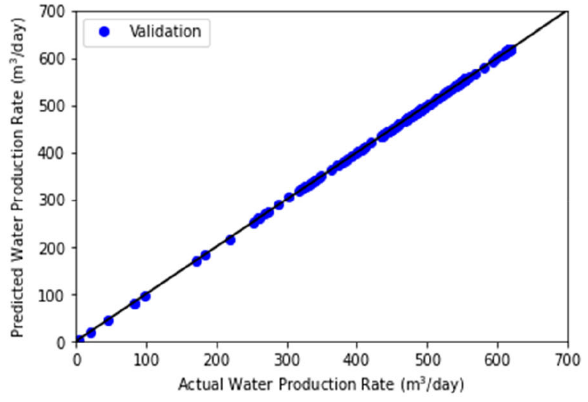
(f)

Figure 10. Water production rate: plots of the correlation coefficient (R^2): for SGD (**a**) training, (**b**) validation, **(c)** testing; for PSO (**d**) training, (**e**) validation, (**f**) testing; and for Adam (**g**) training, (**h**) validation, (**i**) testing.

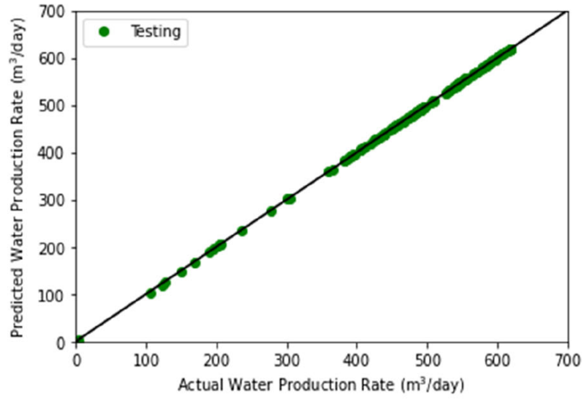
the smart proxy, the dynamic inputs should in fact be estimated by the smart proxy itself. For instance, the smart proxy in this work was developed to predict the oil production rates (also water production rates). This denotes that the oil production rate (water production rates) estimated at the timestep $t-1$ should be used as one of the inputs to approximate the rate at the timestep t. Therefore, if there are more than one outputs to be predicted, then those estimated outputs at the current timestep



(g)

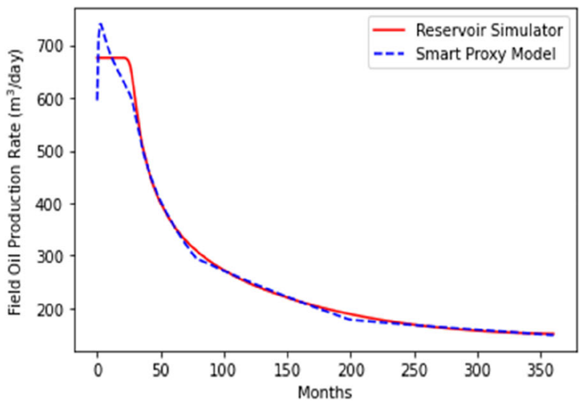


(h)

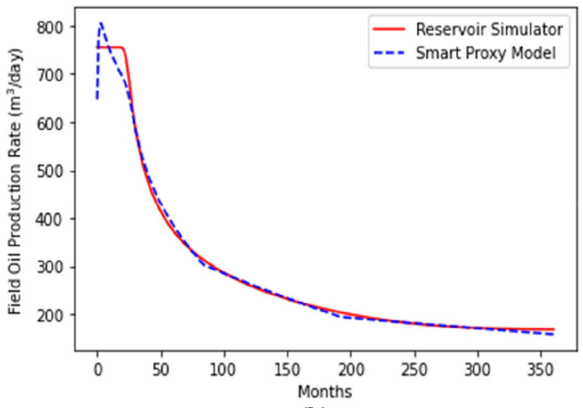


(i)

Figure 10. continued.



(a)



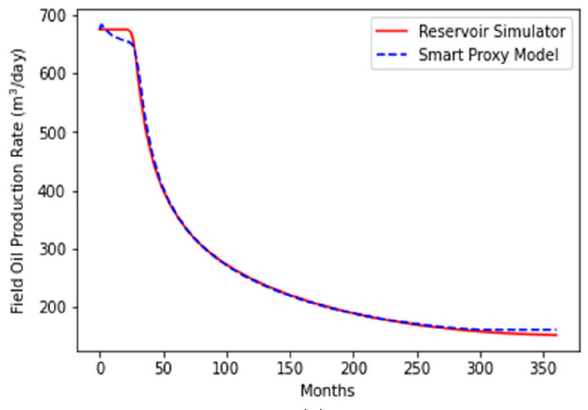
(b)

Figure 11. Oil rate prediction by SGD: plots of the comparison of rates for the results predicted by the trained smart proxy for the two blind cases: (**a**) injection rate of 676 m^3/day; (**b**) injection rate of 755 m^3/day.

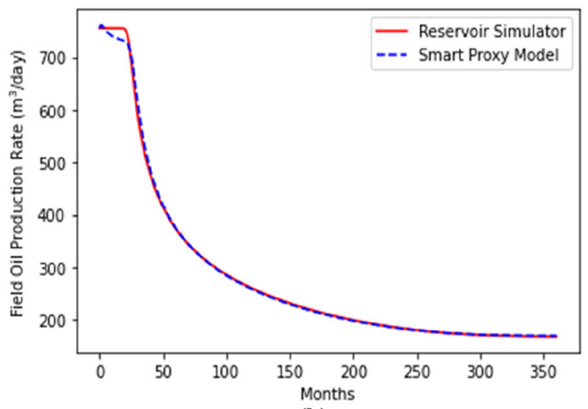
should be cascaded simultaneously to be the inputs at the next timestep. Alternatively, different smart proxy can be designed specifically to provide a prediction of any of the outputs, which is used as the input for another smart proxy. This situation reflects another disadvantage[6] of the application of smart proxy.

Here, only smart proxies that estimated the production rate were developed. For practical and illustrative purposes, other dynamic data, which are used as input data, were retrieved from the reservoir simulation as these data were not used directly in the optimization task discussed. Nevertheless, in this case, the oil production rate estimated by the smart proxy at the current timestep was cascaded to be the input for the approximation of the rate at the next timestep. The plots of the actual output (yielded by reservoir simulator) and the predicted output (produced by SPM) at injection rates of 676 m^3/day and 755 m^3/day are illustrated in Figure 11 for oil rate prediction using SGD, Figure 12 for oil rate prediction using PSO, Figure 13 for oil rate prediction

[6] Building several smart proxies for estimating the dynamic inputs can reduce the convenience of SPM. So, the resolution of this issue will enable a smart proxy to be more tractable.

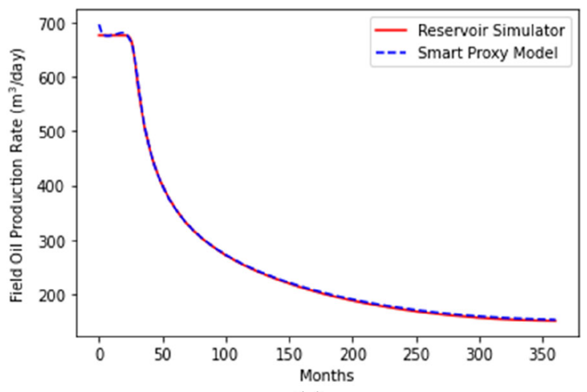


(a)

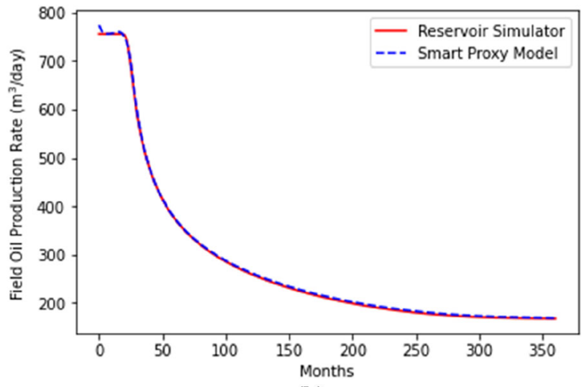


(b)

Figure 12. Oil rate prediction by PSO: plots of the comparison of rates for the results predicted by the trained smart proxy for the two blind cases: (**a**) injection rate of 676 m^3/day; (**b**) injection rate of 755 m^3/day.



(a)



(b)

Figure 13. Oil rate prediction by Adam: plots of the comparison of rates for the results predicted by the trained smart proxy for the two blind cases: (**a**) injection rate of 676 m^3/day; (**b**) injection rate of 755 m^3/day.

using Adam, Figure 14 for water rate prediction using SGD, Figure 15 for water rate prediction using PSO, and Figure 16 for oil rate prediction using Adam. The results of the performance analysis of the two blind cases are presented in Table 10 for oil rate prediction and in Table 11 for water rate prediction. Figures 11, 12, and 13 demonstrate that SGD results in a worse prediction at the beginning of the production (at both targets of injection rate) as compared to PSO and Adam. Despite this, the developed SPMs (trained by both algorithms) for oil rate prediction function were within an acceptable range of accuracy. This is verified by the results shown in Table 10. For water rate prediction, according to Figures 14, 15 and 16, it is explicit that the proxy trained with Adam yielded a better prediction than the models trained with SGD and PSO. However, it is challenging to determine whether PSO was better than SGD. In this case, Table 11 shows that the model trained with PSO predicted better. In this case, the AAPEs resulted from the water rate prediction by using the model trained with SGD were not provided due to the same reason as discussed previously.

In general, when the two blind cases were employed, it was observed that the ANN models trained with any of the three algorithms for both oil and water rates prediction yielded results that are within acceptable range of accuracy. Nevertheless, the performance metrics illustrate that the SPMs built here (for prediction of both oil and water rates) trained by using Adam had a better predictive performance as compared to the models trained by SGD and PSO, whereas PSO outperformed SGD. In addition, we noticed that the SPMs (trained by using both algorithms) in this work had a better prediction of the oil production rates than the prediction of the water production rates. Hence, additional information (e.g., water breakthrough time, total production

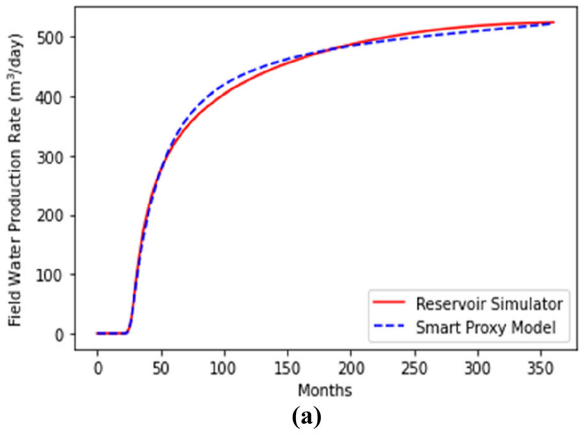


Figure 14. Water rate prediction by SGD: plots of the comparison of rates for the results predicted by the trained smart proxy for the two blind cases: (a) injection rate of 676 m^3/day; (b) injection rate of 755 m^3/day.

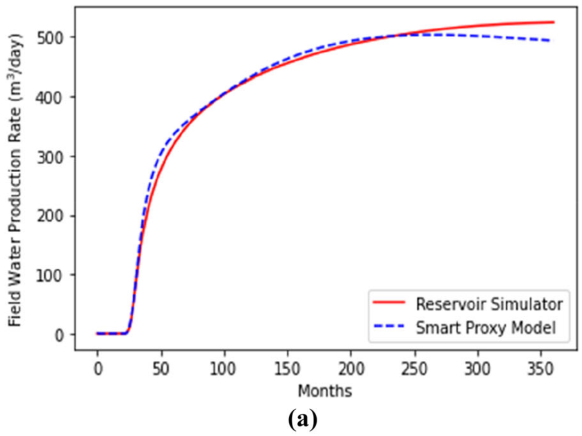
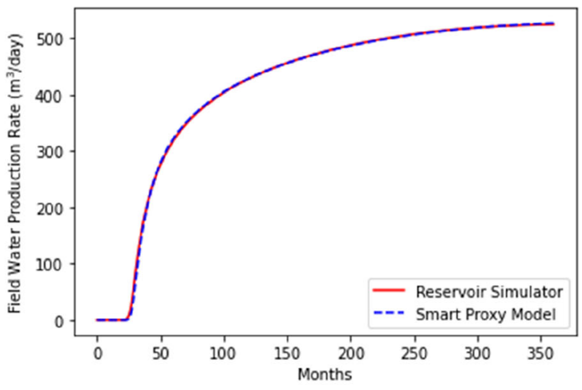


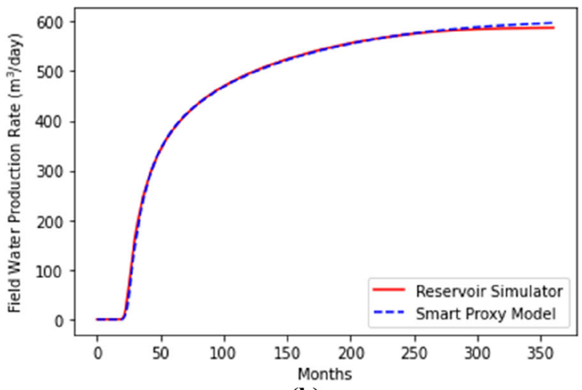
Figure 15. Water rate prediction by PSO: plots of the comparison of rates for the results predicted by the trained smart proxy for the two blind cases: (a) injection rate of 676 m^3/day; (b) injection rate of 755 m^3/day.

of water) can be included as input data to improve the performance of the SPM for water rate prediction.

After obtaining the flow rates predicted by the built SPMs, we proceeded to the illustrative production optimization task. As briefly discussed above, the optimization task here was to select the target of injection rate (between 676 m^3/day and 755 m^3/day) that maximizes the objective function in Eq. (23). By using Eqs. (23) and (24) along with the parameters listed in Table 2, the evolution of NPV throughout the 30 years of production lifetime was determined and is presented in Figure 17. The base cases shown in Figure 17 correspond to the cases for the flow rates derived from the numerical reservoir simulation to determine the evolution of NPV. Both proxy models can reproduce the general trend of the NPV evolution that is close to the one generated by the base cases. This observation is justifiable as all the proxy models yielded the general trends of both oil and water production rates as discussed earlier. Furthermore, from Table 12, all the models reached to the same decision that having the target of injection rate to be 755 m^3/day for 30 years (without termination of production during the period of 30 years) will result in the maximum value of NPV. For the target rate of 676 m^3/day, the percentage error of the NPV resulted from the smart proxy of SGD was about 2.67%, that of PSO was around 1.41%, and that of Adam was about 0.61%. For the target rate of 755 m^3/day, the percentage errors of the NPVs resulted from both proxy models of SGD and PSO were close, namely 1.38% for SGD and 1.33% for PSO. However, for Adam, the percentage error was approximately 0.43%. In this case, the smart proxy trained by using Adam was deemed better. We understand that the economic model used here might be insufficient to reflect the real-life



(a)



(b)

Figure 16. Water rate prediction by Adam: plots of the comparison of rates for the results predicted by the trained smart proxy for the two blind cases: (**a**) injection rate of 676 m^3/day; (**b**) injection rate of 755 m^3/day.

Table 10. Performance metrics of the smart proxy for the two blind cases (oil rate prediction)

	Injection rate	AAPE (%)	RMSE	R^2
Stochastic gradient descent	676 m^3/day	1.849	13.05	0.9924
	755 m^3/day	1.978	13.23	0.9932
Particle swarm optimization	676 m^3/day	1.391	5.701	0.9985
	755 m^3/day	0.708	5.695	0.9988
Adam	676 m^3/day	0.999	2.501	0.9997
	755 m^3/day	1.057	2.830	0.9997

optimization case. However, we aimed to provide insights regarding the use of SPMs in production optimization on a fundamental level.

We also provide a brief discussion on the computational time of these proxy models to highlight the advantage of applying them. The computation here included all the training, validation, testing phases as well as the prediction using the two blind cases. It was done by using a PC with configurations that included Intel® Core™ i9-9900 CPU @3.10 GHz with 64.0 GB RAM. Here, the computation of one of the simulation scenarios listed in Table 3 took about 160 s to finish. When all the five simulation scenarios were run simultaneously, it spent about 290 s to be fully completed. Nevertheless, for the SPM developed here, the computation time of the proxy trained with SGD was about 110 s, that of PSO was about 50 s, and that of Adam was about 120 s.[7] In this aspect, the computation of the proxy trained with backpropagation algorithm was more expensive than that of PSO because PSO is a derivative-free method. In general, we saw that there was still a noticeable (even not very significant) difference in the computational time between the numerical simulation and the proxy models despite the low complexity of the reservoir model used here.

Table 11. Performance metrics of the smart proxy for the two blind cases (water rate prediction)

	Injection rate	AAPE (%)	RMSE	R^2
Stochastic gradient descent	676 m^3/day	–	13.63	0.9917
	755 m^3/day		12.97	0.9935
Particle swarm optimization	676 m^3/day	8.623	8.454	0.9968
	755 m^3/day	7.266	8.975	0.9969
Adam	676 m^3/day	8.049	2.790	0.9996
	755 m^3/day	7.061	4.385	0.9993

Further, we proposed and demonstrated the probabilistic application to investigate further the overall performance of the SPMs. In this case, one of the performance metrics, namely correlation coefficient R^2, was used for illustrative purpose in this part of the work. To do this probabilistic study of the built SPMs, we conducted the process of SPM iteratively for 200 times. This implies that there were 200 samples of R^2 for training phase, validation phase, testing phase, and prediction for each of the two blind cases. Thereafter, the normalized cumulative frequency distribution (NCFD) for R^2 that ranged between 0 and 1 was computed for the 200 samples. In this context, NCFD can be understood as the cumulative number of times for a sample to be within a range of values of R^2 over 200 times. The plots of NCFD are presented in Figures 18, 19, 20, 21, and 22.

[7] Computational time of the proxy built for oil rate prediction was close to that of the proxy developed for water rate prediction.

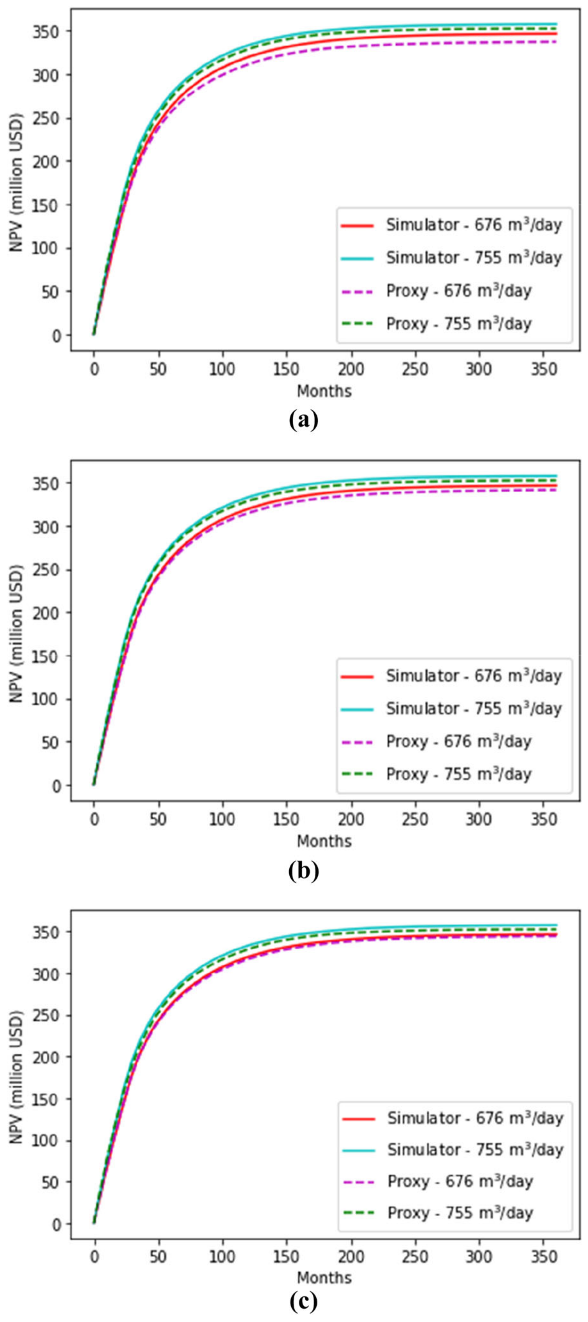


Figure 17. Evolution of NPV throughout the lifetime of production: (**a**) SGD; (**b**) PSO; (**c**) Adam.

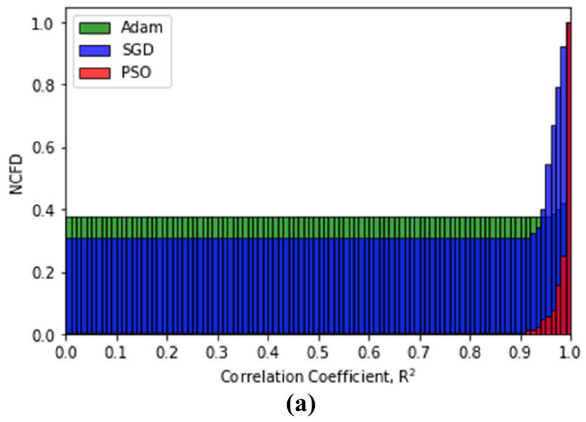
Based on Figure 18, for the training phase of the SPMs, the models trained with PSO had relatively higher chance to result in a healthy training trend than the models trained with the backpropagation algorithms. For the oil rate prediction, PSO had 0.5% chance to result in values of R^2 less than 0.90, whereas SGD had 31% chance and Adam had 37.5% of chance. For the water rate prediction, PSO had about 99% chance to yield values of R^2 that ranged between 0.99 and 1, whereas SGD and Adam, respectively, had only about 60% and 55% chance to achieve that. According to these results, we deduced that PSO was more likely to produce a healthy trend of training compared to SGD and Adam. This deduction is further justified by the results shown in Figure 19 for the validation phase.

For the testing phase, it was noted that the proxy models trained by using PSO performed better that those of SGD and Adam when the models were evaluated against the testing dataset. As portrayed in Figure 20, for the case of oil rate, there was 26% chance that the model trained with PSO will produce values of R^2 less than 0.99 in the testing phase, whereas there was 76% chance that the model trained with SGD will do so; for Adam, the chance was about 47%. Besides, for the case of water rate, PSO had 4% chance to have values of R^2 less than 0.99, whereas SGD had 41.5% and Adam had 45.5%. This provided more confidence that PSO has a higher chance to yield a better predictive performance than SGD when the models were tested with the dataset from a blind scenario.

For the prediction of rates against the datasets from the two blind cases, it can be noticed that, in general, the proxy models by PSO more likely had a better predictive performance than those by SGD and Adam despite the fact that the former had slightly higher chance to produce R^2 values that are less than 0.90 compared with that SGD had in terms of oil rate prediction for injection scenario of 676 m^3/day. This is because based on the prediction of R^2 that ranged between 0.99 and 1, the models by PSO were deemed more reliable than those by SGD and Adam. Besides, in terms of oil rate prediction, Adam statistically had a better chance than SGD in yielding R^2 values between 0.99 and 1 for both injection scenarios. However, for water rate prediction, the chances of both algorithms were very close. We have illustrated that, here, statistically speaking, PSO had a better chance to perform better in training and building the proxy model compared to SGD and Adam. Because PSO is metaheuristics, in

Table 12. Optimal NPVs generated by using all the models

Injection rates	676 m^3/day				755 m^3/day			
models	Simulator	SGD	PSO	Adam	Simulator	SGD	PSO	Adam
NPV$_{optimal}$ (million USD)	346.36	337.11	341.49	344.27	357.35	352.43	352.59	355.84



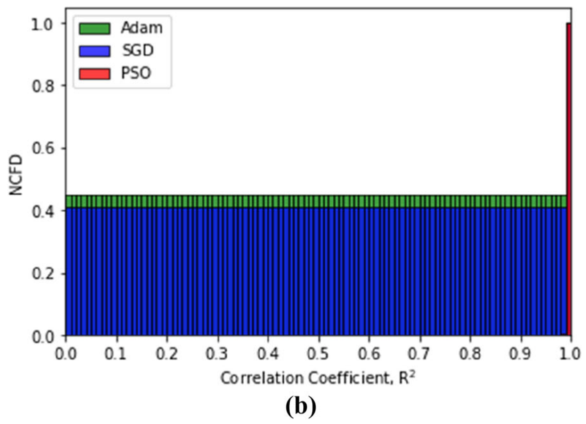
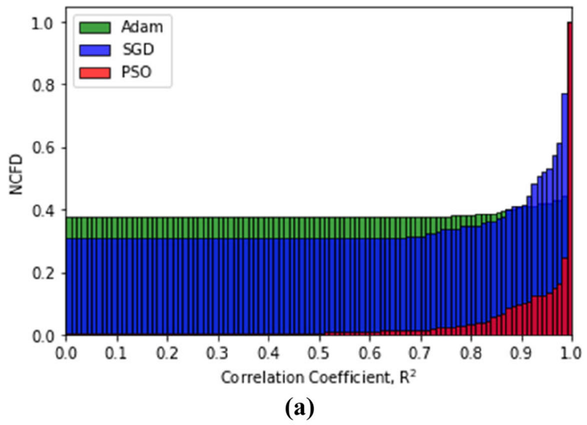


Figure 18. NCFD of R^2 for the training phase of the SPMs: (**a**) oil rate; (**b**) water rate.



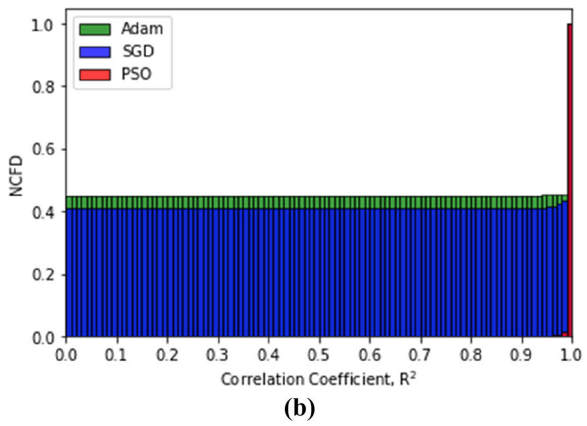
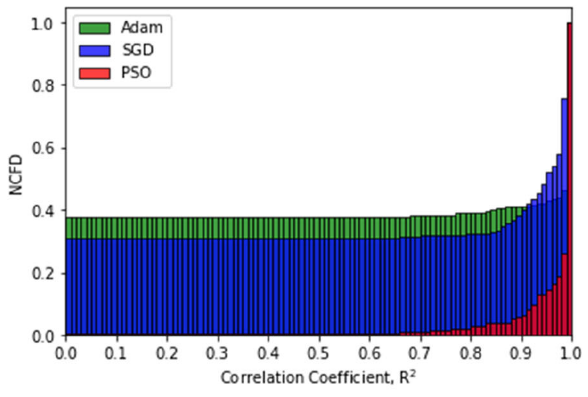


Figure 19. NCFD of R^2 for the validation phase of the SPMs: (**a**) oil rate; (**b**) water rate.

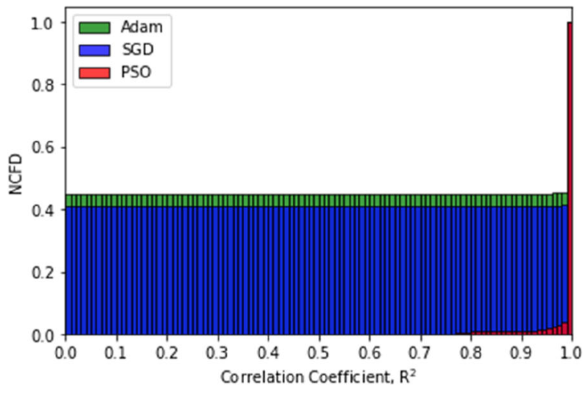
which both global search and local search are balanced, it has a higher chance to have a more exhaustive search in the solution space during the neural network training. Nevertheless, we recommend that this study is conducted using other performance metrics for a more established understanding regarding the outcomes of SPM. Integration of this statistical study in SPM can provide insights about the reliability of an algorithm in training a proxy model and the prediction accuracy of the trained models.

HETEROGENEOUS MODEL

To demonstrate further the robustness of the methodology, we used another fractured reservoir model as a second case study. The general architecture and fluid properties of this new model are similar to those of the previous model. However, we changed the values of some reservoir parameters, including the height of matrix block and the porosity values of both matrix and fracture media, and introduced heterogeneity to the permeability fields

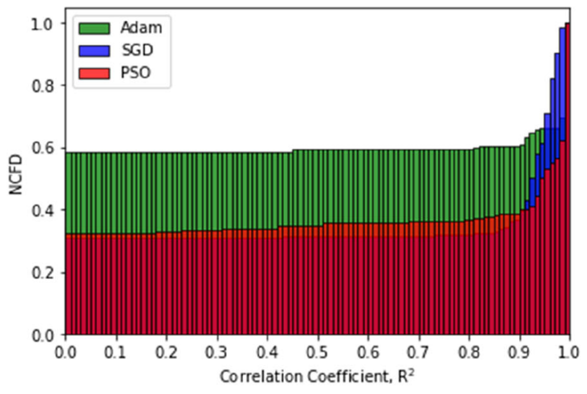


(a)

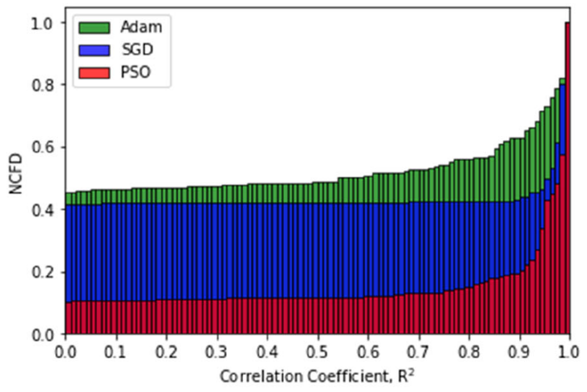


(b)

Figure 20. NCFD of R^2 for the testing phase of the SPMs: (**a**) oil rate; (**b**) water rate.



(a)



(b)

Figure 21. NCFD of R^2 for the prediction of rate of the SPMs when target rate was 676 m^3/day: (**a**) oil rate; (**b**) water rate.

Table 13. Modified reservoir parameters for the heterogeneous model

Layer	Matrix block height (m)	Porosity	
		Matrix media	Fracture media
1	4.572	0.150	0.0050
2	10.67	0.400	0.0020
3	7.620	0.280	0.0015

of both media. In this case, the heterogeneity only applies to permeability. The permeability values in the *x*-, -*y*, and *z*- directions are the same. Thus, the fractured model illustrated here is an isotropic heterogeneous model. Refer to Table 13 for the new values of the heights of matrix blocks and the porosity values. Figure 23 shows the permeability field of each layer in the unit of m^2.

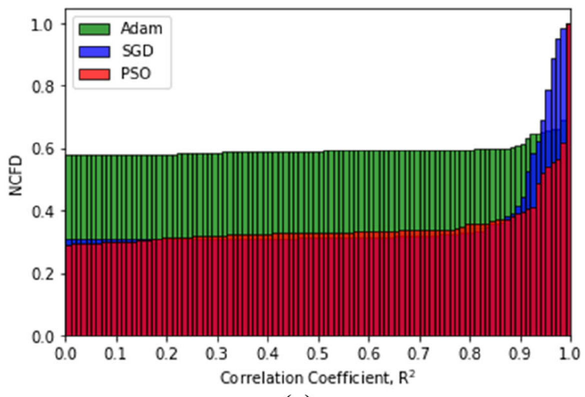
After building this new model by applying the same methodology, the database was extracted and used to develop the SPMs to correspondingly predict the field oil and water production rates. The injection scenarios employed in this case study were the same as in Table 3. The structure of ANN models built here also remained the same as presented in Table 6. This also applied to the use of essential parameters of the three algorithms. For practical and concise purposes, only two performance metrics, namely RMSE and R^2, were implemented to evaluate the training and predictive performance of these proxy models. Table 14 shows the results of training, validation, and testing of the SPM for oil production rate forecasting, whereas Table 15 presents those of the model for water production rate prediction. Generally, the models trained by all the three algorithms yielded excellent training results for both oil and water production rates. Based on

Table 14. Performance metrics of the smart proxy for oil rate prediction based on training, validation, and testing sets

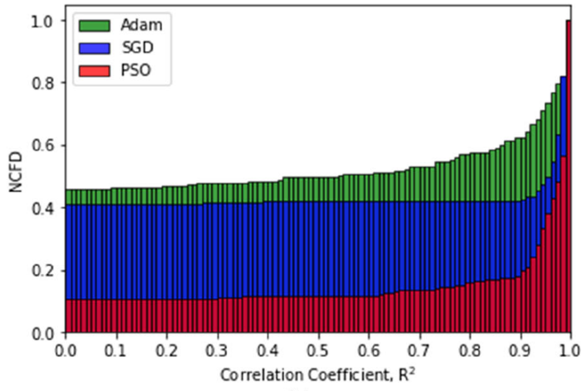
		RMSE	R^2
Stochastic gradient descent	Training (758 data)	7.855	0.9977
	Validation (163 data)	4.700	0.9992
	Testing (162 data)	8.202	0.9978
Particle swarm optimization	Training (758 data)	3.846	0.9995
	Validation (163 data)	3.918	0.9995
	Testing (162 data)	2.739	0.9997
Adam	Training (758 data)	3.154	0.9997
	Validation (163 data)	2.410	0.9998
	Testing (162 data)	3.391	0.9996

Table 15. Performance metrics of the smart proxy for water rate prediction based on training, validation, and testing sets

		RMSE	R^2
Stochastic gradient descent	Training (758 data)	2.401	0.9998
	Validation (163 data)	2.273	0.9998
	Testing (162 data)	2.379	0.9998
Particle swarm optimization	Training (758 data)	1.869	0.9999
	Validation (163 data)	1.961	0.9999
	Testing (162 data)	1.824	0.9999
Adam	Training (758 data)	0.540	0.9999
	Validation (163 data)	0.478	0.9999
	Testing (162 data)	0.422	0.9999



(a)



(b)

Figure 22. NCFD of R^2 for the prediction of rate of the SPMs when target rate was 755 m^3/day: (**a**) oil rate; (**b**) water rate.

both RMSE and R^2, Adam had the best results for both oil and water production rates. Nevertheless, for the testing phase in oil rate proxy model, PSO outperformed the others. For illustrative purposes, only the production profiles estimated by the smart proxies trained by using Adam are presented; the oil profiles are shown in Figure 24, whereas the water profiles are presented in Figure 25.

Thereafter, these models also underwent the blind validation phases by using the two blind cases as explained before. Table 16 records the results of blind validation for oil rate prediction, and Table 17 shows the results for water rate forecasting. For this case study, the PSO outperformed the others when it was used to train the predictive model of oil production rate. However, for the estimation of water production rate, Adam still yielded the predictive model that produced the best results. Then, the production optimization was also done by using the same price setting as shown in Table 2 to highlight the fundamental practicality of the models developed in this case study. The optimal NPVs obtained by using each of the proxy models are tabulated in Table 18.

Based on Table 18, it was deduced that the proxy models built by using Adam produced the optimal NPV with the least percentage error under two different injection scenarios, which were 0.117% for injection rate of 676 m^3/day and 0.329% for injection rate of 755 m^3/day. In addition, all the proxy models reached the same option that the injection rate of 755 m^3/day was economically preferable. Apart from these, for illustrative and succinct purposes, the probabilistic application was only implemented to analyze the predictive performance of each model. The results of this application are demonstrated in Figure 26 for the target rate of 676 m^3/day and in Figure 27 for the target rate of 755 m^3/day. In general, for this case study, it can be

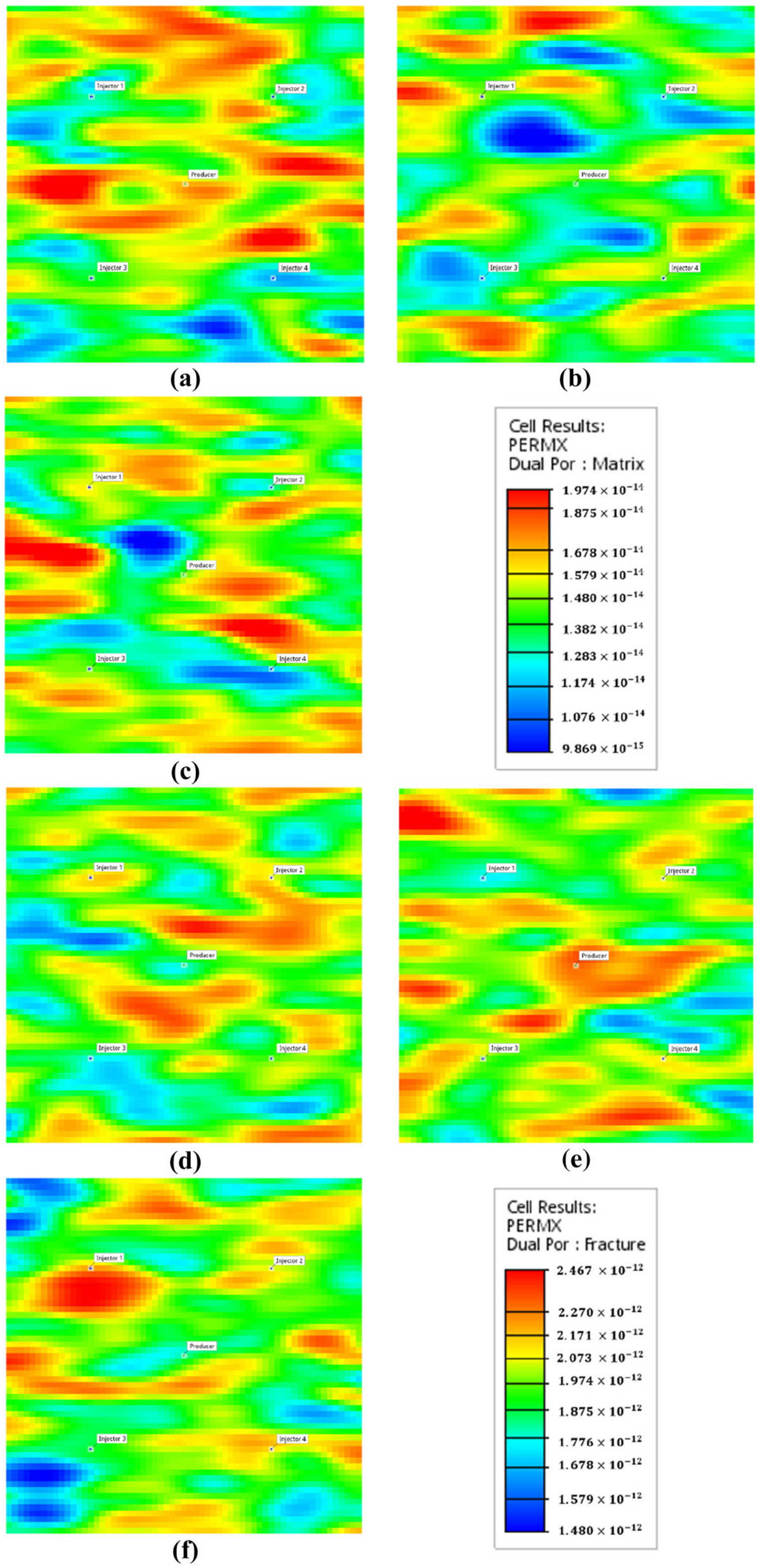
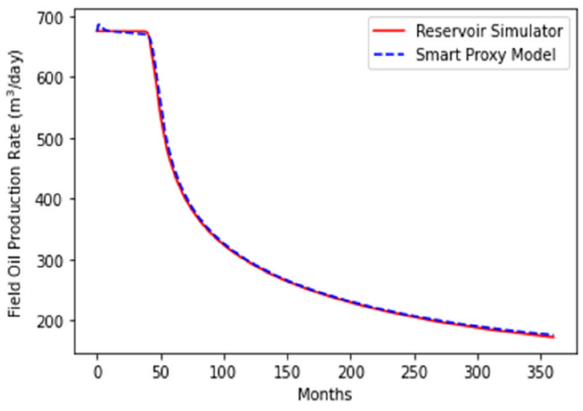


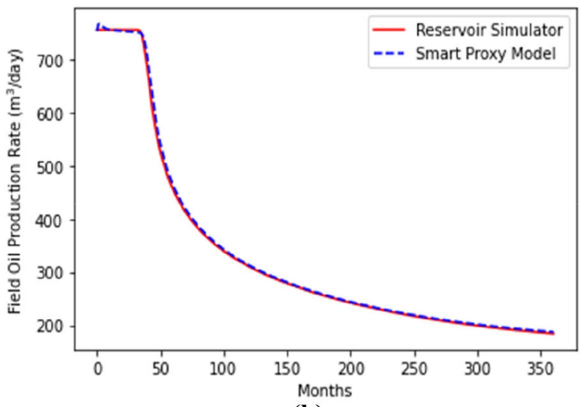
Figure 23. Overview of the isotropic heterogeneous model. The matrix system consists of (**a**) Layer 1, (**b**) Layer 2, and (**c**) Layer 3. The fracture system comprises (**d**) Layer 1, **(e)** Layer 2, (**f**) Layer 3.

Table 16. Oil rate prediction: performance metrics of the smart proxy for the two blind cases

	Injection rate	RMSE	R^2
Stochastic gradient descent	676 m^3/day	12.45	0.9939
	755 m^3/day	13.04	0.9944
Particle Swarm Optimization	676 m^3/day	2.097	0.9998
	755 m^3/day	3.827	0.9995
Adam	676 m^3/day	4.489	0.9992
	755 m^3/day	5.468	0.9990

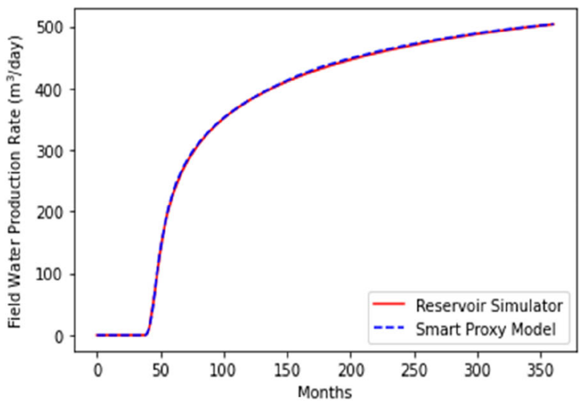


(a)

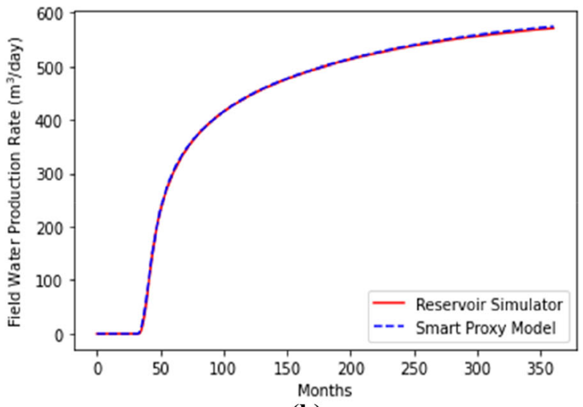


(b)

Figure 24. Oil rate prediction by Adam: plots of the results predicted by the trained smart proxy for the two blind cases: (**a**) injection rate of 676 m^3/day; (**b**) injection rate of 755 m^3/day.



(a)



(b)

Figure 25. Water rate prediction by Adam: plots of the results predicted by the trained smart proxy for the two blind cases: (**a**) injection rate of 676 m^3/day; (**b**) injection rate of 755 m^3/day.

deduced that PSO had a better chance than both SGD and Adam to produce a predictive model with higher accuracy level (i.e., R^2 exceeding 0.99).

CONCLUSIONS

Here, we have shown how SPM can be conducted by using a synthetic fractured reservoir model. The purpose of this study was to provide some insights and a more concrete demonstration regarding the modeling of a smart proxy. We also briefly discussed how the spatio-temporal database can be generated, and we presented the selection of input and output data which were used in the neural network training. This procedure is of paramount importance as a good database determines the success of SPM. Apart from implementing the backpropagation algorithms, namely SGD and Adam, to train the smart proxy, we also demonstrated how the training of a smart proxy can be coupled with PSO. Regarding this, for each training algorithm, we developed two SPMs which correspondingly predicted oil production rate and water production rate. During the development of the smart proxies, all the three algorithms showed excellent training results. However, for the proxy of water rate prediction (trained with both SGD and PSO), some of the resulting AAPEs were large due to the existence of outliers. Despite this, the proxy still showed healthy training and validation trend. In addition, both models illustrated splendid predictive performance as indicated by the results. This shows that the overall predictive performance of the smart proxies remains intact despite having outliers in the neural network training. We consider this as one of the important contributions derived from this work because most of the available literatures solely focus on the use of traditional backpropagation algorithm in SPM. Thereafter, we showed how these SPMs can be used to optimize production through an illustrative example. Besides, we used the performance metrics of correlation coefficient (R^2) for probabilistic evaluation of the overall performance of the SPMs. We summarize our main findings and results derived from this work as follows.

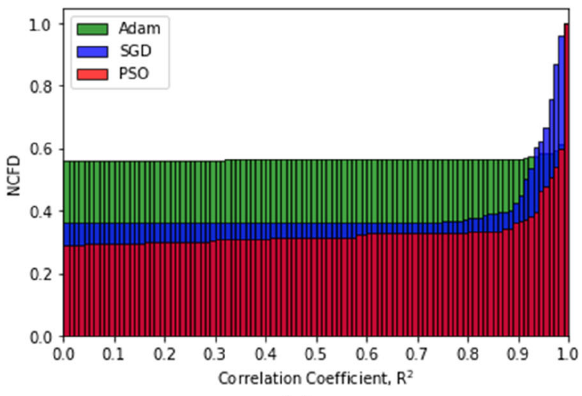
Table 17. Water rate prediction: performance metrics of the smart proxy for the two blind cases

	Injection rate	RMSE	R^2
Stochastic gradient descent	676 m^3/day	5.723	0.9987
	755 m^3/day	10.25	0.9966
Particle swarm optimization	676 m^3/day	9.966	0.9961
	755 m^3/day	7.705	0.9981
Adam	676 m^3/day	1.589	0.9999
	755 m^3/day	1.921	0.9999

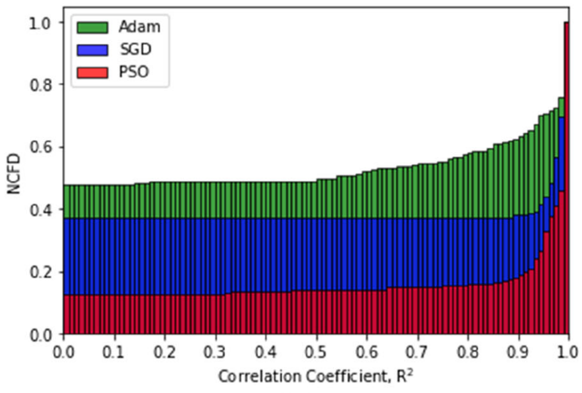
1. Based on the deterministic analysis conducted for SPM of oil rate prediction, the performance metrics (based on training, validation, and testing) showed that Adam generally yielded lower AAPE, RMSE, and higher R^2 than SGD and PSO. However, for the RMSE in the validation phase, PSO resulted in the highest value due to the existence of outliers as previously discussed. Besides, for SPM of water rate prediction, the performance metrics portrayed that Adam was also generally better than SGD and PSO.
2. For oil rate prediction of the blind cases, proxy model with Adam also had the lowest AAPE, RMSE, and the highest R^2. The same results were obtained for water rate prediction.
3. For the production optimization case, the SPMs trained with all three algorithms reached the same decision as what the base case did, which was to select the target injection rate to be 755 m^3/day. However, the NPVs calculated using the data obtained from the proxy model built with Adam were much closer to those estimated by using the data from reservoir simulator.
4. According to the probabilistic analysis for prediction of oil and water rates, it is inferred that PSO has a higher chance to generate a SPM that can result in excellent training and predictive performance compared with SGD and Adam.
5. The same methodology was also applied to an isotropic heterogeneous fractured reservoir model to illustrate its robustness. For this, it was generally found out that Adam can outperform SGD and PSO in the development of the SPMs. However, for oil production rates, PSO produced a better testing result. Regarding blind validation, Adam also generally resulted in more accu-

Table 18. Optimal NPVs generated by using all the models

Injection rates	676 m^3/day				755 m^3/day			
Models	Simulator	SGD	PSO	Adam	Simulator	SGD	PSO	Adam
NPV$_{\text{optimal}}$ (million USD)	428.92	421.77	424.31	429.43	447.22	440.97	444.04	448.69

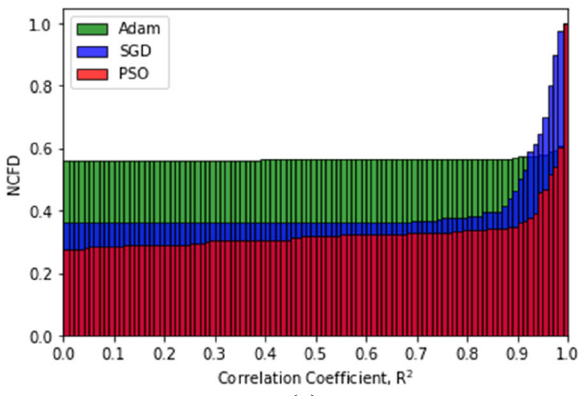


(a)

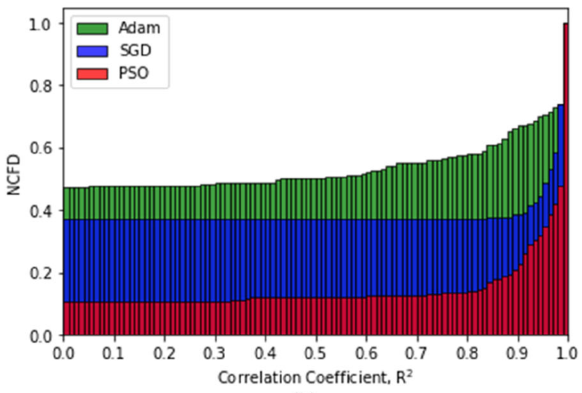


(b)

Figure 26. NCFD of R^2 for the prediction of rate of the SPMs when target rate was 676 m^3/day: (**a**) oil rate; (**b**) water rate.



(a)



(b)

Figure 27. NCFD of R^2 for the prediction of rate of the SPMs when target rate was 755 m^3/day: (**a**) Oil rate' (**b**) Water rate.

rate predictive models of water production rates. Nonetheless, the predictive model of oil rates established by using PSO estimated the oil profile more accurately. Additionally, PSO showed higher chance than SGD and Adam to produce models with excellent predictive ability.

Based on the findings presented, we conclude that, in this work, a metaheuristic algorithm can be applied aptly to train and build a good smart proxy of a fractured reservoir model. Although it has been demonstrated that PSO might not deterministically outperform the considered backpropagation algorithms in smart proxy modeling, statistically it still has a better chance to yield a good performance in this case study. Nonetheless, we understand that there are still some shortcomings regarding these SPMs. We hope that these proxies can be enhanced to be more tractable and robust[8] in terms of prediction of any reservoir-related parameter. In short, we believe that we have achieved the main goals of this work, which include a vivid illustration of SPM, an integration of metaheuristic algorithm in proxy training, a presentation of practical use of the built proxies in optimization on a fundamental level, and an inclusion of a probabilistic application in evaluating a proxy model.

ACKNOWLEDGMENTS

This research is a part of BRU21 – NTNU Research and Innovation Program on Digital Automation Solutions for the Oil and Gas Industry (www.ntnu.edu/bru21). The authors would also like

[8] ‘‘Robust’’ here implies that the smart proxy model can be used in real-life cases.

to acknowledge the support given by the Department of Geoscience and Petroleum (IGP) of NTNU to this research work.

OPEN ACCESS

FUNDING

Open access funding provided by NTNU Norwegian University of Science and Technology (incl St. Olavs Hospital - Trondheim University Hospital).!/FundingInformation>

REFERENCES

Ahmad, S. A., & Olivier, R. G. (2008). Matrix-fracture transfer function in dual-medium flow simulation: review, comparison, and validation. In *Europec/EAGE Conference and Exhibition*. Rome, Italy. https://doi.org/https://doi.org/10.2118/113890-MS.

Alenezi, F., & Mohaghegh, S. D. (2016). A data-driven smart proxy model for a comprehensive reservoirsimulation. In *The 4th Saudi International Conference on Information Technology (Big Data Analysis) (KACSTIT)*. Riyadh, Saudi Arabia. https://doi.org/10.1109/KACSTIT.2016.7756063.

Alenezi, F., & Mohaghegh, S. D. (2017). Developing a smart proxy for the SACROC water-flooding numerical reservoir simulation model. In SPE Western Regional Meeting. Bakersfield, California, United States. https://doi.org/https://doi.org/10.2118/185691-MS.

Barrenblatt, G. I., Zheltov, I. P., & Kochina, I. N. (1960). Basic concepts in the theory of homogeneous liquids in fissured rocks. *Journal of Applied Mathematics and Mechanics, 24*(5), 1286–1303.

Bianchi, L., Dorigo, M., Gambardella, L. M., & Gutjahr, W. J. (2009). A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing, 8,* 239–287.

British Petroleum. (2020). BP Statistical Review of World Energy 2020. https://www.bp.com/en/global/corporate/energy-economics/statistical-review-of-world-energy.html. Retrieved July 21, 2020.

Buduma, N., & Locasio, N. (2017). *Fundamentals of deep learning: Designing next-generation machine intelligence algorithms*. Sebastopol, California, United States: O'Reilly.

Ceetron Solution AS. (2020). *ResInsight*.

Chollet, F. (2019). *Keras, version 2.3.0*.

Ertekin, T., & Sun, Q. (2019). Artificial intelligence applications in reservoir engineering: A status check. *Energies, 12*(15), 2897.

Firoozabadi, A., & Thomas, L. K. (1990). Sixth SPE comparative solution project: Dual-porosity simulators. *Journal of Petroleum Technology, 42*(6), 710–715.

Gerald, P., Raftley, A. E., Ševčíková, H., Li, N., Gu, D., Spoorenberg, T., et al. (2014). World population stabilization unlikely this century. *Science, 346*(6206), 234–237.

Gharbi, R. B. C., & Mansoori, G. A. (2005). An introduction to artificial intelligence applications in petroleum exploration and production. *Journal of Petroleum Science and Engineering, 49*(3–4), 93–96.

Gilman, J. R., & Kazemi, H. (1983). Improvements in simulation of naturally fractured reservoirs. *Society of Petroleum Engineers Journal, 23*(4), 695–707.

Google Brain Team. (2020). *TensorFlow, version 2.1.0*.

He, Q., Mohaghegh, S. D., & Liu, Z. (2016). Reservoir simulation using smart proxy in SACROC unit—case study. 2016. In *SPE Eastern Regional Meeting*. Canton, Ohio, United States. https://doi.org/https://doi.org/10.2118/184069-MS.

International Energy Agency. (2018). World Energy Outlook 2018. International Energy Agency. https://www.iea.org/weo2018/. Retrieved September 24, 2019.

Kazemi, H., Merrill, L. S., Jr., Porterfield, K. L., & Zeman, P. R. (1976). Numerical simulation of water-oil flow in naturally fractured reservoirs. *Society of Petroleum Engineers Journal, 16*(6), 317–326.

Kennedy, J., & Eberhart, R. (1995). Particle Swarm Optimization. In *IEEE International Conference on Neural Networks*. Perth, Western Australia, Australia. https://doi.org/10.1109/ICNN.1995.488968.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In *The 3rd International Conference for Learning Representations*. San Diego, California, United States. arxiv.org/abs/1412.6980.

Lescaroux, F., & Mignon, V. (2009). On the influence of oil prices on economic activity and other macroeconomic and financial variables. *OPEC Energy Review, 32*(4), 343–380.

Mohaghegh, S. D. (2000a). Virtual Intelligence applications in petroleum engineering: Part 1—artificial neural networks. *Journal of Petroleum Technology, 52*(9), 64–72.

Mohaghegh, S. D. (2000b). Virtual intelligence applications in petroleum engineering: Part 2—evolutionary computing. *Journal of Petroleum Technology, 52*(10), 40–46.

Mohaghegh, S. D. (2000c). Virtual intelligence applications in petroleum engineering: Part 3—fuzzy logic. *Journal of Petroleum Technology, 52*(11), 82–87.

Mohaghegh, S. D., Modavi, C. A., Hafez, H. H., Haajizadeh, M., Kenawy, M.M., & Guruswamy, S. (2006). Development of surrogate reservoir models (SRM) for fast track analysis of complex reservoirs. In *Intelligent Energy Conference and Exhibition*. Amsterdam, The Netherlands. https://doi.org/https://doi.org/10.2118/99667-MS.

Mohaghegh, S. D. (2011). Reservoir simulation and modeling based on pattern recognition. In *SPE Digital Energy Conference and Exhibition*. The Woodlands, Texas, USA. https://doi.org/https://doi.org/10.2118/143179-MS.

Mohaghegh, S. D., Liu, J. S., Gaskari, R., Mayasami, M., & Olukoko, O. A. (2012a). Application of surrogate reservoir models (SRM) to an onshore green field in Saudi Arabia; Case Study. In *North Africa Technical Conference and Exhibition*. Cairo, Egypt. https://doi.org/https://doi.org/10.2118/151994-MS.

Mohaghegh, S. D., Amini, S., Gholami, V., Gaskari, R., & Bromhal, G. S. (2012b). Grid-based surrogate reservoir modeling (SRM) for fast track analysis of numerical reservoir simulation models at the gridblock level. In SPE Western Regional Meeting. Bakersfield, California, United States. https://doi.org/https://doi.org/10.2118/153844-MS.

Mohaghegh, S. D., Liu, J. S., Gaskari, R., Mayasami, M., & Olukoko, O. A. (2012c). Application of surrogate reservoir models (SRM) to Two offshore fields in Saudi Arabia; Case Study. In *SPE Western Regional Meeting*. Bakersfield, California, United States. https://doi.org/https://doi.org/10.2118/153845-MS.

Mohaghegh, S. D., Abdulla, F., Abdou, M., Gaskari, R., & Mayasami, M. (2015). Smart proxy: An innovative reservoir management tool; Case study of a giant mature oilfield in the UAE. In *Abu Dhabi International Petroleum Exhibition and Conference*. Abu Dhabi, UAE. https://doi.org/https://doi.org/10.2118/177829-MS.

Mohaghegh, S. D. (2017). *Data-Driven Reservoir Modeling*. Richardson, Texas, United States: Society of Petroleum Engineers.

Mohaghegh, S. D. (2018). *Data-driven analytics for the geological storage of CO_2*. Boca Raton, Florida, United States: CRC Press.

Miranda, L. J. V. (2019). *PySwarms, version 1.1.0*.

Nait Amar, M., Zeraibi, N., & Redouane, K. (2018). Bottom hole pressure estimation using hybridization neural networks and grey wolves optimization. *Petroleum, 4*(4), 419–429.

Nait Amar, M., Zeraibi, N., & Redouane, K. (2018). Pure CO_2-oil system miscibility pressure prediction using optimized neural network by differential evolution. *Petroleum and Coal, 60*(2), 284–293.

Nait Amar, M., & Jahanbani Ghahfarokhi, A. (2020). Prediction of CO2 diffusivity in brine using white-box machine learning. *Journal of Petroleum Science and Engineering, 190,* 107037.

Nait Amar, M., Zeraibi, N., & Jahanbani Ghahfarokhi, A. (2020). Applying hybrid support vector regression and genetic algorithm to water alternating CO2 gas EOR. *Greenhouse Gases: Science and Technology, 10*(3), 613–630.

Parada, C. H., & Ertekin, T. (2012). A New screening tool for improved oil recovery methods using artificial neural networks. In *SPE Western Regional Meeting*. Bakersfield, California, United States. https://doi.org/https://doi.org/10.2118/153321-MS.

Python Software Foundation. (2020). *Python version 3.8.1.*.

Romm, E. S. (1966). *Fluid flow in fractured rocks*. Moscow, Russia: Nedra Publishing House. (**in Russian**).

Schlumberger. (2020a). *ECLIPSE 100 Reservoir Engineering Software*.

Schlumberger. (2020b). Technical Challenges - Carbonate Reservoirs; https://www.slb.com/technical-challenges/carbonates.

Shi, Y., & Eberhart, R. (1998). A modified particle swarm optimizer. In *IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*. Anchorage, Alaska, United States. https://doi.org/10.1109/ICEC.1998.699146.

Shi, Y., & Eberhart, R. (1999). Empirical study of particle swarm optimization. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*. Washington DC, United States. https://doi.org/10.1109/CEC.1999.785511.

Su, S., Gosselin, O., Parvizi, H., and Giddins, M. A. (2013). Dynamic matrix-fracture transfer behavior in dual-porosity models. In *EAGE Annual Conference and Exhibition incorporating SPE Europec*. London, UK. https://doi.org/https://doi.org/10.2118/164855-MS.

Uleberg, K., & Kleppe, J. (1996). Dual porosity, dual permeability formulation for fractured reservoir simulation. In *RUTH Seminar*. Stavanger, Norway. http://www.ipt.ntnu.no/~kleppe/TPG4150/fracturedpaper.pdf.

Van Golf-Racht, T. D. (1982). *Fundamentals of fractured reservoir engineering*. New York, United States: Elsevier Scientific.

Warren, J. E., & Root, P. J. (1963). The behavior of naturally fractured reservoirs. *Society of Petroleum Engineers Journal, 3*(3), 245–255.

Wiggins, M. L., & Startzman, R. A. (1990). An approach to reservoir management. In *SPE Annual Technical Conference and Exhibition 1990*. New Orleans, Louisiana, USA. https://doi.org/https://doi.org/10.2118/20747-MS.

Wilamowski, B. M., & Irwin, J. D. (2011). *The industrial electronics handbook second edition: Intelligent system*. Boca Raton, Florida, United States: CRC Press.

Zhang, Y., Wang, S., & Ji, G. (2015). A comprehensive survey on particle swarm optimization algorithm and its applications. *Mathematical Problems in Engineering, 2015,* 1–38.