Neural Computing and Applications (2023) 35:16633–16647
https://doi.org/10.1007/s00521-023-08537-6

ORIGINAL ARTICLE

Actor-critic reinforcement learning leads decision-making in energy systems optimization—steam injection optimization

Ramez Abdalla[1,2] · Wolfgang Hollstein[1] · Carlos Paz Carvajal[1] · Philip Jaeger[1]

Received: 18 November 2022 / Accepted: 22 March 2023 / Published online: 27 April 2023

Abstract
Steam injection is a popular technique to enhance oil recovery in mature oil fields. However, the conventional approach of using a constant steam rate over an extended period can lead to sub-optimal performance due to the complex nature of the problem and reservoir heterogeneity. To address this issue, the Markov decision process can be employed to formulate the problem for reinforcement learning (RL) applications. The RL agent is trained to optimize the steam injection rate by interacting with a reservoir simulation model and receives rewards for each action. The agent's policy and value functions are updated through continuous interaction with the environment until convergence is achieved, leading to a more efficient steam injection strategy for enhancing oil recovery. In this study, an actor-critic RL architecture was employed to train the agent to find the optimal strategy (i.e., policy). The environment was represented by a reservoir simulation model, and the agent's actions were based on the observed state. The policy function gave a probability distribution of the actions that the agent could take, while the value function determined the expected yield for an agent starting from a given state. The agent interacted with the environment for several episodes until convergence was achieved. The improvement in net present value (NPV) achieved by the agent was a significant indication of the effectiveness of the RL-based approach. The NPV reflects the economic benefits of the optimized steam injection strategy. The agent was able to achieve this improvement by finding the optimal policies. One of the key advantages of the optimal policy was the decrease in total field heat losses. This is a critical factor in the efficiency of the steam injection process. Heat loss can reduce the efficiency of the process and lead to lower oil recovery rates. By minimizing heat loss, the agent was able to optimize the steam injection process and increase oil recovery rates. The optimal policy had four regions characterized by slight changes in a stable injection rate to increase the average reservoir pressure, increasing the injection rate to a maximum value, steeply decreasing the injection rate, and slightly changing the injection rate to maintain the average reservoir temperature. These regions reflect the different phases of the steam injection process and demonstrate the complexity of the problem. Overall, the results of this study demonstrate the effectiveness of RL in optimizing steam injection in mature oil fields. The use of RL can help address the complexity of the problem and improve the efficiency of the oil recovery process. This study provides a framework for future research in this area and highlights the potential of RL for addressing other complex problems in the energy industry.

Keywords Physics informed reinforcement learning · Production optimization · Reservoir management · Digital transformation in oil and gas industry

✉ Ramez Abdalla
rm31@tu-clausthal.de

Wolfgang Hollstein
wolfgang.hollstein@tu-clausthal.de

Carlos Paz Carvajal
carlos.paz@tu-clausthal.de

Philip Jaeger
jaeger.philip@tu-clausthal.de

[1] Clausthal University of Technology, Clausthal, Germany

[2] Datagration Solutions Inc., Baden, Austria

1 Introduction

In general, the recent technological advancements in upstream fields make closed-loop reservoir management approaches popular and attractive solutions. Production optimization is considered the ultimate goal of closed-loop reservoir management [1]. It is done by using updated reservoir model(s), optimal well controls (such that the NPV of the production is maximized), or hydrocarbon recovery in the reservoir lifecycle is maximized [2].

Steam flooding is a thermal oil recovery method. In this method, steam forms a condensing zone inside the reservoir. The heat of condensation is utilized to heat up the heavy crude oil, facilitating its displacement due to a reduction in viscosity [3]. The injected steam forms a steam chamber around the injection well [4]. Figure 1 shows the steam chamber around the injection well. This steam chamber is expanded toward the production well. Consequently, the condensed water displaces the reservoir fluid into the production well [4, 5].

Because of the high steam cost, one of the necessary objectives in steam injection projects is the optimization of the dynamic injection rates. Due to the reservoir simulation complexity involved in the steam injection, no single steam injection rate value can be optimal for all steam flooding situations. Also, it is difficult for engineers to find a strategy to inject steam over the entire production horizon that maximizes the net present value. Hence, there are some attempts in the literature to optimize steam injection rates. These attempts can be divided into two groups. The first group deals with steam injection as a steady-state problem, which means such research would provide only one single value of injection rate as an optimum value along the production horizon. The other group studies the problem as a dynamic optimization problem, where the output would be an injection rate strategy over time.

Both optimization solutions would have the objective of maximizing the cumulative performance of the wells. The first group includes evolutionary algorithms such as non-dominated sorting genetic algorithms and particle swarm optimization algorithms to optimize injection rate. Amei et al. have applied particle swarm optimization, a genetic algorithm, and an imperialist competitive algorithm in order to optimize the steam-to-oil ratio [6]. They concluded that the genetic algorithm (GA) worked 6% better in comparison to other optimization techniques and was also faster in comparison to the continuous optimization algorithm. As aforementioned, the biggest drawback of that approach is that it provides only one single value for steam injection or steam ration, which is not sufficient in such a complex problem.

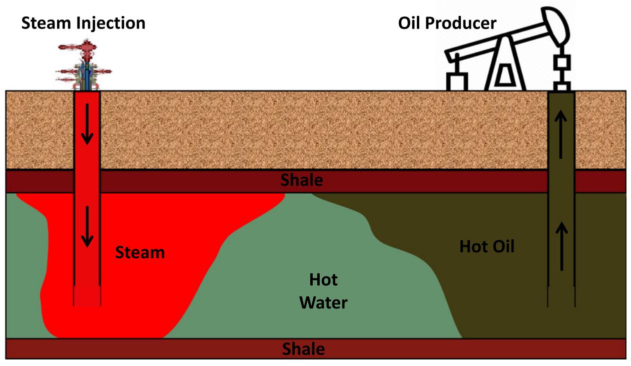


Fig. 1 Steam injection process

On the hand, the second group of researches deal with steam injection as an optimal control problem. This group of researches include the model predictive control algorithm and the adjoint state method. Model Predictive Control (MPC) is considered the most widely used advanced control method in refining, chemical, and petrochemical processes [7–11]. MPC is a control algorithm that is built on the concept of moving horizon [12]. For the steam injection problem, a proxy model is introduced to formulate the problem. This proxy model is a relation between injection rates and oil and water production rates. Finally, the formulated problem is optimized over the prediction horizon to find the steam injection rates that maximize the net present value. However, MPC approach is preferable but the drawback of combining with proxy model to formulate the system makes it unstable and only suitable for small horizon of production [13].

In the second group also lies the adjoint-method. It is perform some sort of gradient-based optimization method where the derivatives are obtained through the use of an adjoint equation or co-state equation [14]. It depends on the barrier function to formulate the augmented objective function. Hence, the augmented objective function includes the calculation of net present value and the constraints that formulate the reservoir dynamics. The disadvantage of this approach is that the equations representing the gradient of the augmented objective function with respect to the steam injection rate must be hard-coded which is not easy in case of thermal oil recovery processes where compositional models are used [15].

2 Reinforcement learning applications in oil and gas industry

More recent applications in the area of complex control included the application of reinforcement learning (RL) [16–18]. It is one of three basic machine learning paradigms, alongside supervised learning and unsupervised learning. Recently, Machine learning applications are rapidly developed in upstream field. It transforms our ability to describe complex systems from observational data, rather than first-principles modeling.

It is obvious that reinforcement learning is different from unsupervised learning. While the latter focuses on the extraction of patterns and useful information hidden in the unlabeled data, reinforcement learning has the ability to map inputs to an output. Although both supervised and reinforcement learning work on mapping inputs to outputs, in reinforcement learning, a learner and decision-maker called an agent is trained to find the optimal policy, which acts as a feedback to the agent and gives RL an advantage in optimization problems. Hence, both supervised and unsupervised have different objectives they are applied for. While supervised learning can be used for solving nonlinear prediction and classification problems [19–24], reinforcement learning is used for data-driven optimization [25–28].

The application of machine learning techniques, particularly reinforcement learning, has shown great promise for dynamic optimization tasks in recent years. The primary objectives for such approaches are typically two-fold: (1) to develop control policies that enable effective mapping from sensor inputs to actuation outputs and (2) to support decision-making in real-time, even in complex and changing environments [29]. Subsequently, machine learning is well-suited to deal with high-dimensional, nonlinear optimization problems that are difficult to model explicitly. Reinforcement learning, in particular, enables agents to learn from their interactions with the environment and adapt their behavior accordingly, making it a powerful tool for control and decision-making in complex systems.

For data-driven optimization, evolutionary algorithms could also be used, but as aforementioned, they are search meta-heuristic algorithms that reflect the process of natural selection, where the fittest individuals are selected for reproduction in order to produce offspring of the next generation, while reinforcement learning is structured as an agent interacting with an environment.

Defining a certain optimization task, RL is concerned with a specific type of optimization problem in which the objective is to find policies (strategies) that maximize the return while an agent interacts with an environment in time steps. On the other hand, evolutionary algorithms are self-learning algorithms that can be applied to any optimization problem where you can encode solutions, define a fitness function that compares solutions, and stochastically change those solutions.

Regarding the problems that each algorithm can solve, RL is concerned with a specific type of optimization problem in which the objective is to find policies (strategies) that maximize the return while an agent interacts with an environment in time steps. On the other hand, evolutionary algorithms are self-learning algorithms that can be applied to any optimization problem where you can encode solutions, define a fitness function that compares solutions, and stochastically change those solutions. Therefore, reinforcement learning would be more suitable in our case to find the optimal policy for injection rates over time.

In this study, we will focus on reinforcement learning applications in closed-loop reservoir management. In general, RL applications are still very few and limited in engineering processes [18, 30, 31] as the respective algorithms are still in their development phases. Also, the research community is still working out the kinks in how advancements are validated and shared [32]. In the energy domain specifically, the systems are too complex to be simulated, and it is also too risky to allow the algorithms to interact directly with production systems [33]. However, RL algorithms may have a greet rule in the further development of artificial intelligence. Therefore, the interest in these algorithms has been continuing for the past decade [34]. Some researchers addressed the application of reinforcement learning in the energy domain. Such applications mainly had objectives regarding decision management or flow control. Field applications are mainly categorized into two groups: either to control autonomous drilling or to optimize production.

In the context of autonomous drilling control [35–37], the objective is tracking some set points; therefore, it is often either a penalty function for downhole pressure in the case of managed pressure drilling or for landing position, final inclination, and maximum curvature in the case of controlled directional drilling positioning.

In the context of production optimization, the general configuration of closed-loop reservoir simulation consists of two main parts as follows: (1) model-based data assimilation, which acts as reservoir parameters and states an estimator, and (2) a model-based optimizer. As mentioned above, the task of the optimizer is to maximize the oil recovery factor or other desired economic criteria such as the NPV. The required inputs for the optimizer part may be field total metrics such as injection and production data or volumetric data from grids such as pressure and water saturation in wells' windows. The objectives are often to maximize production, minimize losses, increase net present value and reduce lost time. Therefore, all of these applications used net present value as a reward function for their agent to search for the optimum policy. However, they have various objectives on the reservoir simulation prospective.

For waterflooding optimization projects, there are three attempts [38–40]. These attempts differ in either the state definition or the complexity of the model or the applied reinforcement learning algorithm. Both Hourfar et al. and Ma et al. [38, 39] have used the estimation of the residual oil in percentages and also the values of water cuts in the total produced fluid. On the other hand, Miftakhov et al. [40] have used pixels of the reservoir simulation model to define the state. It works fine in this case, may be because the model was so simple, but we expect such a state definition would lead to the curse of dimensionality because of the huge number of inputs in such a case.

For steam-assisted gravity drainage, unlike SARSA that has been used by Guevara et al. [15], we propose the usage of the actor-critic reinforcement learning algorithm. It combines the advantages of value-based and policy-based approaches. In this algorithm, the actions are generated by a policy neural network. This network is evaluated based on a corresponding change in state potential. The values of the state potential are given by another neural network called the critic network. This network approximates the expected cumulative reward from the given state. The actor-critic algorithm is also recommended for a competitive non-Markovian environment where a stochastic action can be preferable to a deterministic one. It is also used in the case of continuous action spaces like continuous robotic control [41].

There are other miscellaneous applications such as the usage of reinforcement learning to optimize injection rates in carbon dioxide storage [42] unlike the traditional methods and evolutionary algorithms that are used in the literature [43, 44], the agent could maximize the reward that is built depend on risk and economical factors. There are also other applications in optimal well placement and well type selection in the area of pressure transient tests interpretation [45–47].

In this study, we present a novel approach to production optimization in reservoir simulation through the application of reinforcement learning. Our focus is specifically on the challenges associated with data-driven optimization, which differ from those of data-driven modeling. By prioritizing this concentration, we highlight the key message of our work: that optimizing production through data-driven techniques requires a distinct set of considerations and approaches. We believe that this emphasis on the unique challenges of data-driven optimization is crucial to advancing autonomous drilling and production optimization. Our work represents a significant contribution to this field.

In this work, model-free reinforcement learning (RL) is applied for steam injection rate optimization. It was selected because it overcomes the shortcomings of the aforementioned methods in two ways. Firstly, it does not require a full description of the process required to be optimized. Secondly, this approach takes advantage of previous experiences or interactions with the environment to find an optimal policy of injection rate to maximize the net present value without human interference. Also, among reinforcement learning algorithms, we explore the usage of actor-critic (A2C). The advantages of such an algorithm are the following [48]:

1 Sample efficiency: Actor-Critic is more sample efficient than some other reinforcement learning algorithms, such as Q-learning and SARSA. This is because it uses both the value function and the policy to learn, which can help to reduce the number of interactions needed with the environment.

2 Convergence to an optimal policy: Actor-Critic has a strong theoretical basis for convergence to an optimal policy, and it guarantees improvement at each iteration. This means that the performance of the policy will improve over time.

3 Handling of non-stationary environments: Actor-Critic can handle non-stationary environments where the distribution of the data changes over time. This is because the policy is updated using its own experience, and the critic's value function is updated using the temporal difference error.

4 Ability to balance exploration and exploitation: Actor-Critic can balance exploration and exploitation using the policy and value function. The policy is used to explore the environment and discover new solutions, while the value function is used to exploit the current knowledge of the environment and make decisions based on the expected return of each action.

3 Elements of RL

The reinforcement learning technique is the preferred method for the purpose of this paper. It is highlighted because it is the optimal policy (a policy that maximizes the reward of the decision process). Hence, it supports automated transformation and goes beyond conventional approaches to solve the task at hand. Reinforcement learning problems involve learning how to map situations to actions in a closed-loop structure where the learning system's actions influence its later inputs, which converge toward the maximum numerical reward signal. Moreover, the learner discovers which actions yield the best reward by trying them out. Figure 2 presents agent-environment interaction in RL.

Achieving agent-environment interaction in real life requires four basic features to be well defined. They are the environment, the reward function, the value function, and the policy function with its relevant actions.

3.1 Environment

"Environment" is a mathematical model that mimics the behavior of the environment. In other words, it presents inferences about how the environment behaves. For instance, the model might predict the resultant next state s_{t+1} and next reward r_{t+1} given a state s_t and taking action a_t.

3.2 Reward function

A reward signal is considered the target or the objective function in a reinforcement learning problem. It is a single number sent from the environment to the reinforcement learning agent at any time, depending on the agent's current action and the current state of the environment. The agent can influence the reward signal through its action. Therefore, maximizing the total reward that the agent

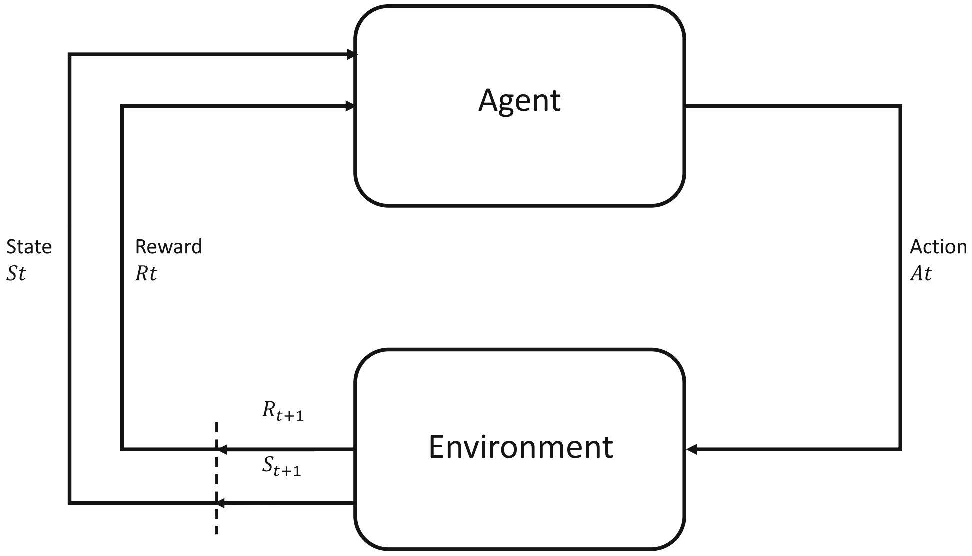


Fig. 2 The agent-environment interaction in RL

receives over the long run is considered its main objective. The reward signal defines good and bad actions for the agent. It is denoted by R_t at time step (t).

3.3 Value function

A value function is used to evaluate success in the long run. Roughly speaking, the value of a state, i.e., the status of the agent (what has been learned, what must be done), is the total amount of rewards an agent can expect to accumulate over the future, starting from a specific state at time (t), which can, for instance, be the initial conditions of the simulations toward the end of the episode. Whereas a reward determines the immediate return at a specific environmental state, a function value indicates the long-term desirability of the state after considering the states that are likely to follow. In other words, the value function at state (s) considers the reward available in (s) and the upcoming states. The value function is denoted by V_π, where $V_\pi(S)$ is the value function for a specific state (S) under a specific policy π.

3.4 Policy function

A policy defines the learning agent's way of behaving at a given time. Roughly speaking, a policy can be considered as an interconnection between states (the input vector from the environment at a specific time step t) and actions (what should be done?). The policy is denoted π_t, where $\pi_t(a \mid s)$ is the probability of taking an action (a) when being in a certain state (s).

4 Learning dynamics (agent-environment interactions)

Over a sequence of discrete time steps $t = \{0, 1, 2, 3, \cdots, T\}$, an agent interacts with an environment E. In every iteration, the agent receives a current state from the environment (st) and selects an action from some set of possible actions (a) according to its policy π. In return, the agent receives the next state s_{t+1} and receives a scalar reward r_t. The process continues until either the agent reaches a terminal state or the maximum number of time steps satisfied after which the process restarts [49].

The total accumulated return from time step t can be calculated from $R_t = \sum_{k=0}^{\infty} \gamma^k\ r_{t+k}$ were $\gamma \in [0, 1]$ is the discount factor. The value of state s under policy π is defined as $V_\pi(s) = [R_t \mid s_t = s]$ and is simply the expected return for following policy pi from state s. Another important function while learning is the action value function $Q_\pi(s, a) = \mathbb{E}[R_t \mid s_t]$. It is the expected cumulative reward an agent can receive by taking a specific action in a specific state, and following a certain policy π.

Solving a reinforcement learning task means reaching the optimal policy. It is the one that going to give us the maximum value in any state we are in $\pi^* = \text{argmax}_\pi[\mathbb{E}_\pi \sum_{k=0}^{\infty} \gamma^k\ r_{t+k}]$. There are two techniques to solve the problem in model-free reinforcement learning either value-based technique or policy-based technique.

In value-based model-free reinforcement learning methods, There is no explicit policy is stored but only solving the problem of the value function [50]. The policy is here implicit and can be derived directly from the value function by picking the action with the best value. For instance DQN is the most popular Q-learning algorithm. In this algorithm, the action value function is represented using a function approximation, such as a neural network $Q(s,\ a;\ \theta)$. The objective is to directly approximate the optimal action value function: $Q*(s,\ a)\ \approx\ Q(s,\ a;\ \theta)$. The learning is done through iterative continuous update of the parameters θ of the action value function minimizing a sequence of loss functions. Equation 1 shows the loss function applied per iteration i.

$$J(\theta_i) = \mathbf{E}(r + \gamma_{a'}^{\max}\ Q(s', a', \theta_{i-1}) - Q(s, a, \theta_i))^2 \tag{1}$$

where s', is the state at $t+1$ after state s at time t.

On the contrary, we explicitly build a parameterized representation of the policy $\pi(a_t \mid s_t; \theta)$ in the policy-based methods. Then, gradient ascent on $\mathbf{E}[R_t]$ is performed to update the parameters θ. An example of such a method is the REINFORCE family of algorithms [51].

In this algorithm the policy parameters θ is updated in the direction $\nabla\theta \log \pi(a_t \mid s_t;\ \theta) * R_t$. In order to solve the problem of the high variance of such estimates while keeping it unbiased, a learned estimate of the value function is commonly used as the baseline $\mathrm{bt(st)}\ \approx\ V\pi(\mathrm{st})$. It is commonly known as a baseline. This base line is subtracted from the return. Hence, the resulting gradient is calculated from $\nabla\theta \log\ \pi(\mathrm{at} \mid \mathrm{st};\ \theta)\ (\mathrm{Rt}\ -\ \mathrm{bt(st)})$ [52].

The advantage actor-critic reinforcement learning combines both techniques, the value based and the policy based, together [51]. It consists of two neural networks and the advantage function. The later calculates the agent's TD Error or Prediction Error. The actor network can be considered as a policy gradient algorithm that chooses an action at each time step. On the other hand, the critic network evaluates the Q-value or provide a feedback on how to adjust. While the critic network learns which states are better or worse, the actor uses the critic results to teach the agent to seek out good states and avoid bad states.

Hence, the combination of deep learning with RL would be a new powerful tool. It is called deep reinforcement learning. Such a combination hindered classical RL by allowing the use of high-dimensional state spaces and exploiting the feature extraction capabilities of deep neural networks. That makes deep reinforcement learning suitable to be explored in the field of fluid mechanics where challenging environments with nonlinear problems of high dimensionality are confronted. Hence, if neural networks are properly trained, they can estimate value function and output an action distribution given set of states and therefore represents arbitrarily complex mappings between spaces [53].

5 Steam injection model

Reservoir simulation is a crucial tool for understanding the behavior of oil and gas reservoirs, as it allows engineers to predict the flow of fluids through porous media over time. However, simulating a reservoir can be a highly complex task, particularly when dealing with fluids such as gas condensate and volatile oil.

The models used for simulating these types of reservoirs are known as compositional models. They typically involve a complex mix of thermodynamic relations and equations for modeling the differential material balances, which must take into account the phase transitions of the different occurring components. In other words, conservation laws are applied for each component in each grid, using equations for fluid transport through porous media, rock or fluid properties, and the conservation of mass, momentum, and energy in order to solve the problem.

This results in a large system of highly nonlinear differential equations, which must be solved for each grid block using computers. To achieve this, the reservoir must first be discretized into a large number of grid blocks to describe the dynamic behavior of a real hydrocarbon reservoir. However, because real reservoirs are often highly heterogeneous in terms of pore structure, oil saturation, and other factors, a significant number of grid blocks must be defined to accurately capture the behavior of the system.

Furthermore, optimizing the production of hydrocarbons in a reservoir involves significant complexity. The goal of optimization is to determine the optimal well placement and production schedule to maximize the recovery of hydrocarbons while minimizing costs and environmental impact. This involves a range of factors, including reservoir characterization, wellbore construction, fluid properties, and surface facilities.

Optimization strategies typically involve using simulation models to evaluate different scenarios and determine the most effective production strategy. However, this can be a complex and time-consuming process, as it involves running multiple simulations with different input parameters and evaluating the results to determine the optimal solution. Overall, the complexity of reservoir simulation and optimization highlights the importance of using advanced computational tools and strategies to improve the efficiency and accuracy of hydrocarbon production.

In this use case, the reservoir simulation model represents the environment where oil displacement is carried out using steam injection. Specifically, an inverted nine-spot pattern is used [54], which is divided into eight equal parts, with one of the parts being modeled. The total area of the pattern is 2.5 acres, and a grid system of $23 \times 12 \times 12$ is used to represent the reservoir. This means that the reservoir is divided into a large number of small grid blocks, with the grid points distributed uniformly in the horizontal plane. The well radii for all three wells are 0.3 ft, which is an important parameter in determining the effectiveness of the steam injection process. The case is shown in Fig. 3.

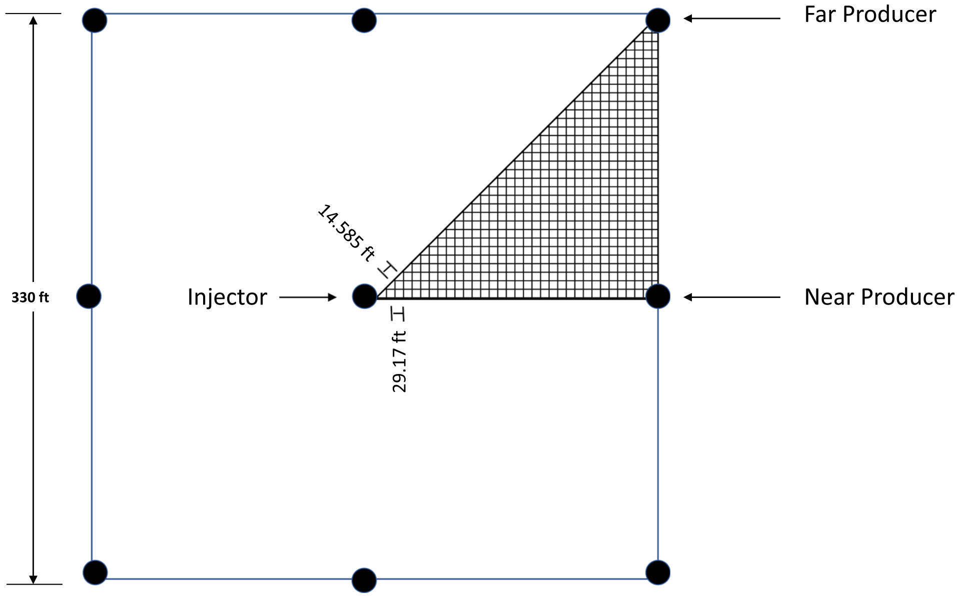


Fig. 3 3 element of symmetry used in the simulation of steam injection in an inverted nine-spot

Figure 3 shows element of symmetry used in the simulation of steam injection in an inverted nine-spot. Rock Properties: Vertical permeability is 50% of horizontal values. Porosity of all layers is 0.3 (fraction). Thermal conductivity of reservoir overburden and under-burden is 24 BTU/(ft-D-OF). Heat capacity of reservoir overburden and under-burden is 35 Btu/(ft3 of rock-OF). Effective rock compressibility is 5 × 10-4 psi- 1.

Fluid Properties: Properties of pure water are assumed Oil density at standard conditions is 60.68 Ibm/ft3. compressibility is 5×10-6 psi-1. The coefficient of thermal expansion is 3.8 × 10-4 OR-1, and the molecular weight is 600. Temperature and viscosity data are shown in Table 1. Further information about the produced fluid properties and initial oil composition can be found in Tables 2 and 3.

Rock-fluids interaction: For water/oil system, the residual oil saturation (Sorn) is 0.15. For gas/oil system, the residual oil saturation (Sorg) is 0.10. The critical gas saturation (Sgc) is 0.06.

Table 1 Temperature and viscosity data

Temperature (°F)	Viscosity (cp)
75	5780
100	1380
150	187
200	47
250	17.4
300	8.5
350	5.2
500	2.5

Table 2 Properties of oil components

	Components		
	1	2	3
Molecular weight	250	450	600
Specific heat, Btu/Ibm-oR	0.53	0.55	0.6
Density at standard conditions Ibm/ft3	52.3	57.64	61.2
Critic pressure, psia	225	140	–
Critical temperature, °F	800	950	–

Table 3 Initial oil composition

Components	Mole fraction
C1	0.5030
C2	0.1614
C3	0.3356

Regarding permeability, the oil relative permeability at interstitial water saturation, kroiw is 0.4. For water/oil system, water relative permeability at residual oil saturation (krwro) is 0.1. For gas/oil system, gas relative permeability at residual oil saturation (krgro) is 0.2. Equations (2)–(5) define the relative permeability for the water/oil system and for the gas/oil system as

$$k_{rw} = k_{rwro}\left(\frac{S_w - S_{wir}}{1 - S_{orw} - S_{wir}}\right)^{2.5} \quad (2)$$

$$k_{row} = k_{roiw}\left(\frac{1 - S_{orw} - S_w}{1 - S_{orw} - S_{iw}}\right)^{2} \quad (3)$$

$$k_{rg} = k_{rgro}\left(\frac{S_g - S_{gc}}{1 - S_{iw} - S_{gc}}\right)^{1.5} \tag{4}$$

$$k_{rw} = k_{rwro}\left(\frac{1 - S_{iw} - S_{org} - S_g}{1 - S_{iw} - S_{org}}\right)^{2} \tag{5}$$

Initial Conditions The initial oil and water saturation is 55% and 45% respectively. Reservoir temperature is 125°F. The pressure at the center of the top layer is 75 psia.

Operating Conditions The steam is injected into the bottom layer only, while production occurs in all four layers. Steam injection capacity is subject to the following constraints: (I) a maximum BHP of 1600 psia at the center of the bottom layer and (II) a maximum injection rate of 300 STBID on a full-well basis. The capacity of the production wells is subject to the following constraints: (I) a minimum BHP of 17 psi at the center of the top layer; (II) a maximum production rate of 1000 STBID of liquids. The operation shall take 820 days of injection and production.

The next step would be to reformulate the reservoir simulation problem to match the Markov decision process (MDP). After preparing the model, the actor-critic agent starts to interact with the environment for a production period of 820 days. Multiple agents interacting with the reservoir simulation environment as a multi-agent actor-critic can be considered a next level of study. It is also worth mentioning that this study is based on the general assumption of a reliable history-matched model.

6 Steam Injection problem formulation based on RL

6.1 State

As aforementioned, a true MDP state should provide something that is Markovian and captures the environment at its fundamental level. Therefore, the state of the reservoir simulation should carry enough information to capture the history of the process or the previously applied injection rates. Equation 6 shows how the state is set at every time step.

$$\begin{aligned} St = [&\text{Cumulative oil production},\\ &\text{Cumulative steam injection},\\ &\text{Cumulative water production}] \end{aligned} \tag{6}$$

6.2 Actions

"In this study, a discrete action space consisting of three possible actions is used: increase, decrease, and no change in steam injection rate. The action taken in each time step is with reference to the previous time step injection rate by a constant value (20 BPD). This design of the action space prevents dramatic changes in steam injection rate." The action space for the agent is designated according to Eq. 7.

$$a_t = \begin{cases} Q_{\text{inj}}(t-1) + 20 & \text{if action} == 0 \\ Q_{\text{inj}}(t-1) - 20 & \text{if action} == 1 \\ Q_{\text{inj}}(t-1) & \text{if action} == 2 \end{cases} \tag{7}$$

Table 4 Economical factor

Parameter	Value
P_{oil}	100
C_{steam}	12
C_{water}	3
$i_{[\text{fraction}]}$	0.2

6.3 Reward function

During the reinforcement learning process, the reward function serves as an essential guide for the agent. In this study, the net present value (NPV) of the reservoir, as defined by Eq. 8, is utilized as the metric for the reward function. The ultimate aim is to optimize the steam injection strategy to maximize the cumulative sum of NPVs calculated at each discrete time step. This serves as a key indicator of the agent's efficacy in enhancing oil recovery through the use of steam injection.

$$R_t = \text{NPV}_t = \sum_{n=1}^{N} \frac{P_o q_o - C_{\text{steam}}\, q_s - C_{\text{water}} q_w}{1 + i^{\frac{t - t_{\text{ref}}}{365}}} \tag{8}$$

where

- $P_{\text{oil}}, C_{\text{steam}}$ *and* C_{water} are the oil price, the cost of steam generation and the cost of produced water handling in [USD/STB].
- q_o, q_s *and* q_w are the oil production rate, steam injection rate and water production rate in [STB/day].
- t, *and* t_{ref} are the current time and the reference time to which NPV is discounted.
- i is the annual discount factor. The economical parameters are presented in Table 4.

6.4 Components summary

A task, an environment and an agent are the main elements of reinforcement learning optimization technique. For our study, The contextual interpretations of these elements is summarized in Table 5.

Table 5 Elements of the reinforcement learning in the context of steam injection

Element	Steam injection context
Environment	Compositional models of reservoir simulation
Task	To find optimal injection steam injection rate autonomously along 820 days of production
State	Operating conditions St = [Cumulative oil production, Cumulative steam injection, Cumulative water production]
Reward	Net present value
Agent	To send actions to the injector well
Policy	To find optimal policy of steam injection rate to maximize the net present value
Action	Manipulating parameter of the operating conditions which is the injection rate

6.5 Implementation actor to critic method

The model consists of three wells (one injector and two producers), and a production horizon of 820 days (one episode) is considered. The state of the environment is defined as cumulative, as oil and water production, and as steam injection. For each time step, the three possible actions defined previously are considered, and the reward is represented by the NPV.

The training process is conducted through successive interactions between the agent and the environment. The agent executes an action each time step according to a specific policy, and then it receives a reward that is the net present value in our case. Hence, the policy is improved through the action taken and the observation of new states of the environment with each interaction. Then, the agent learns to gain rewards so that it acts correctly in situations not present in the training set.

In SARSA, the agent interacts with the environment and updates the policy based on actions taken. SARSA learns the Q-values associated to the policy that itself follows $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t))$. Hence, this approach converges with possible penalties from exploratory moves but at the same time it avoids a possibly dangerous optimal path at the cost of learning slowly. Subsequently, SARSA is rather preferred in robotic experiments and less in simulations in order to avoid damages.

Instead, in the actor-critic method, the actor proposes a set of possible actions given a determined state, i.e., the actor assumes the function of the policy (where to go?) and the critic, on the other side, evaluates the actions taken by the actor. This evaluation is defined as the "estimated value function."

Using data from a reservoir, the environment is built as a simulation model. Figure 4 shows the agent (A2C)-environment (reservoir simulator) interaction in our implementation for the determination of optimal operating conditions

In this study, the agent can be inferred to represent an injector well. It provide the environment with actions that result in the optimal operating conditions for the simulated system. This agent is trained based on the net present value that is calculated using the feedback from the simulator. The agent used is an actor-critic. The actor network represents a parameterized policy (π_θ). Hence, it is responsible for mapping states (st) to actions (at). The output of the actor's network is a 3-dimensional vector representing the probability of the three actions. Those are increase, decrease and not changing in steam injection rate with reference to the previous injection rate. Then, using the probability distribution presented in Eq. 9, the action is determined. On the other hand, the critic network evaluates the impact of actions by estimating the Q-value of a state-action pair. Hence, it takes both the state and the action as inputs. This ensures that the actor to critic agent consistently makes the best decision.

$$\pi_\theta(s,a) = \arg\max(\mathrm{softmax}\, p[a \mid s, \theta]) \tag{9}$$

The actor agent is trained through the policy gradient method and optimized policy parameters are obtained through iterations. Equations 10 and 11 show the policy gradient cost function and parameters update.

$$\nabla_\theta J(\theta) = \nabla_\theta \log_{\pi_\theta}(a \mid s) V^{\pi_\theta}(S_t) \tag{10}$$

$$\theta_{t+1} = \theta_t + \alpha\, \nabla_\theta\, J(\theta) \tag{11}$$

Combining Equations (10 and 11) with the value function equation 12 results in Eq. 13

$$V_\pi(S) = \sum_{a \in A} \pi(a,s)(R_s^a + \gamma \sum_{s' \epsilon\, S} T_{ss'}^a V^\pi(s')) \tag{12}$$

$$\theta_{t+1} = \theta_t + \alpha[\nabla_\theta \log_{\pi_\theta}(a|s) \sum_{a \in A} \pi(a,s)(R_s^a + \gamma \sum_{s' \in S} T_{ss'}^a V^\pi(s')] \tag{13}$$

Such a technique exhibits high variance since trajectories can lead to different returns. This is mainly due to the stochasticity of the environment (random events during episodes) and the stochasticity of the policy. For instance, the same starting state can lead to very different returns. Because of this, the return at a specific point starting from the same state can vary significantly across episodes.

During iterations, the trajectory update was observed to have a high variance due to stochasticity of the environment and stochasticity of the policy. For instance, the same starting state can lead to different returns. Therefore, the return starting at the same state can vary significantly across episodes. The solution to mitigate the aforementioned problem is the usage of an advantage function Eq. 14. The advantage function captures the degree of importance an action has compared to others for a given state, while the value function judges the strength of the decision.

$$\begin{aligned} A^{\pi_\theta}(S,a) &= Q^{\pi_\theta}(S,\ a) - V^{\pi_\theta}(S) \\ &= r(a_t, S_t) + \gamma\ V^{\pi_\theta}(S_{t+1}) - V^{\pi_\theta}(S) \end{aligned} \tag{14}$$

Using Eq. 13 and 14, we get equations of actor and critic weights update 15 and 16

$$\theta_{t+1} = \theta_t + \alpha[\nabla_\theta log_{\pi_\theta}(a \mid s)A^{\pi_\theta}(S,a)] \tag{15}$$

$$W_{t+1} = W_t + \beta A[\nabla_w V^{\pi_\theta}(S_t)] \tag{16}$$

Algorithm 1 presents pseudocode for the actor-critic algorithm for the steam injection process.

policy, we will compare it to a baseline policy where the steam injection rate is kept constant at 138 bbl/day.

By analyzing the learning curve, we gain a deeper understanding of the agent's behavior and performance. Figure 5 shows the learning curve for our agent, where the line represents the cumulative net present value (NPV) for each episode after 820 days. As observed, there are large variations in the NPV of initial episodes, which can be attributed to poor approximations of the actor network. In reinforcement learning, the critic typically employs a state-value function. After each actor selection, the critic evaluates the new state to determine whether things have improved or become worse than expected. This evaluation is based on the temporal difference (TD) error, which measures the difference between the predicted and actual values of the state-value function.

As the number of episodes increases, the critic learns about the policy being followed by the actor. This critique takes the form of a TD error, which serves as the sole output of the critic and drives all learning in both actors and critics. By minimizing the TD error through gradient descent, the actor network is trained to select actions that maximize the expected return. The critic network, on the other hand, is updated to better estimate the state-value function, which in turn improves the accuracy of the TD error signal.

```
Algorithm 1 Actor Critic algorithm for the steam injection process
Require: MDP formulation of the reservoir model
Require: Actor network: policy improvement, actions
Require: Critic network: policy evaluation, value function
Ensure: Networks parameters are set: learning rate, number of neurons and
    discount factor
1: for episode in Episodes do
2:     Reset the environment, returns $S_t$ (first state of the episode
3:     for step in Steps do
4:         Execute an action$S_t, a_t,\ R_t,\ a_{t+1}, \pi(a_t \mid S_t)$
5:         Agent get the environment state $S_t, a_t,\ R_t,\ S_{t+1}$
6:         Compute advantage $A^{\pi_\theta}\ (S,a)$
7:         Update policy parameter $\theta_{t+1}$ in actor network
8:         Update value function parameter $W_{t+1}$ in critic network
9:         Update action and state
10:    end for
11: end for
```

7 Results of the steam injection optimization

In this section, we will perform a quasi-experimental study to evaluate the performance of an AI-based agent in optimizing the steam injection rate over time. Specifically, we are interested in examining the learning curve that the agent exhibits and the optimal steam injection rate policy that it obtains. To assess the effectiveness of the agent's

Figure 6 shows the optimal steam injection curves for the a2c agent and the base model versus time. Four specific regions can be distinguished in the optimal injection policy: from 0 to 380 days, the injection rate has slightly changed between 98 to 118 bbl/day. The second region is from 380 to 640 days where injection rate increased from

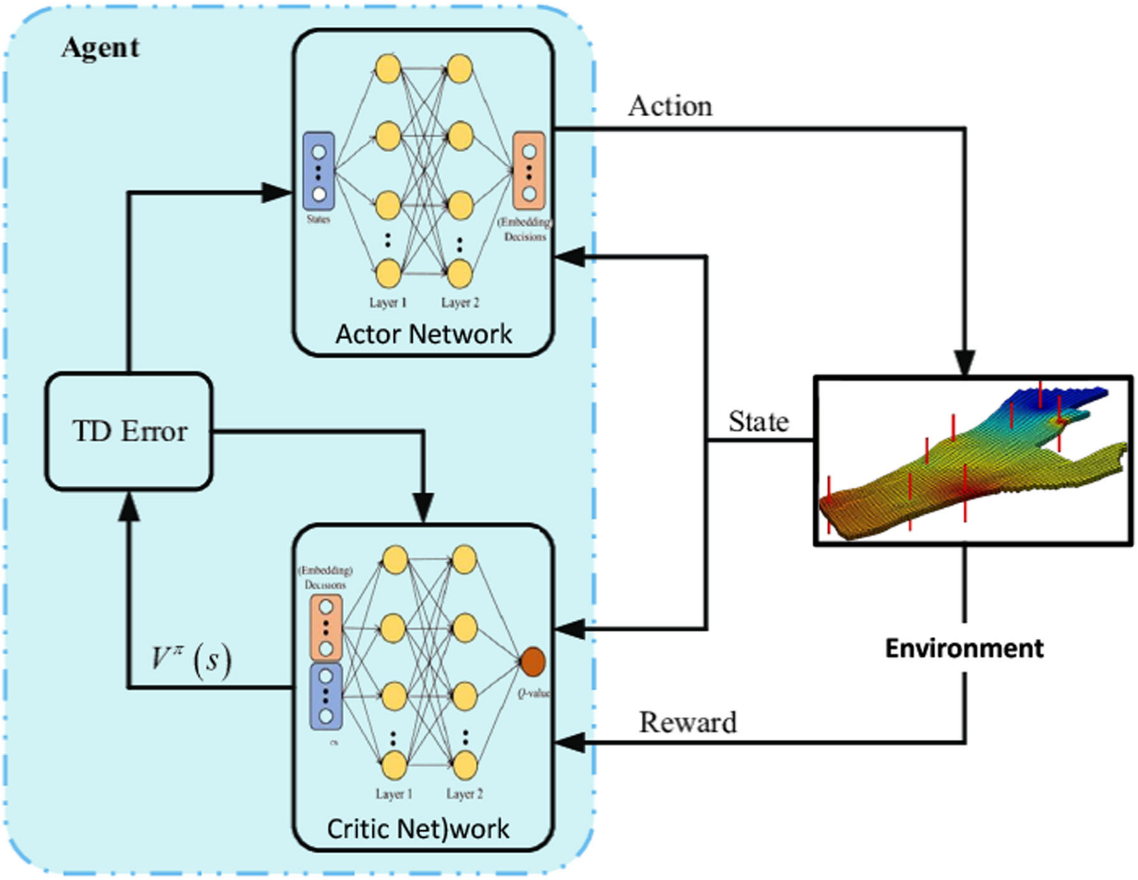


Fig. 4 Actor-critic architecture and its interaction with the environment

118 bbl/day to a maximum value 148 bbl/day. The third region (from 640 to 780) shows a steep decrease in the injection rate from 148 bbl/day to 98bbl/day. The fourth is from 780 to 820 showing a slight change between 98 bbl/day and 108 bbl/day. As per Fig. 6 and from economical perspective, it is clear that the policy defined by the agent was able to achieve the highest net present value. However, the question that arises here is how can such a policy be justified from a physical point of view.

The main aspect that explains the success of this policy is that its ability to minimize the cumulative heat loss to the surrounding rocks. Fig 7a, b shows the rate of heat loss and

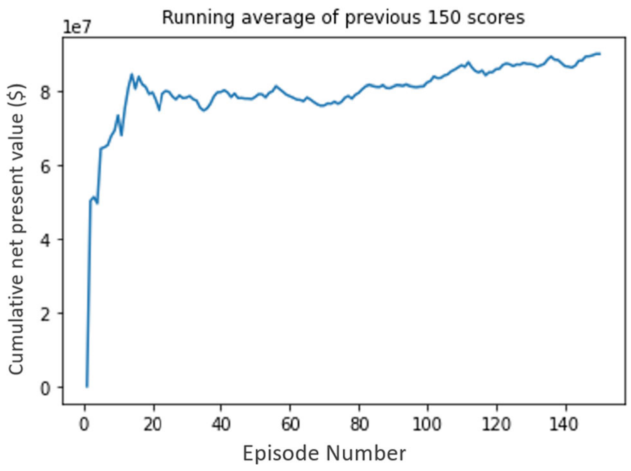


Fig. 5 Learning curve of RL

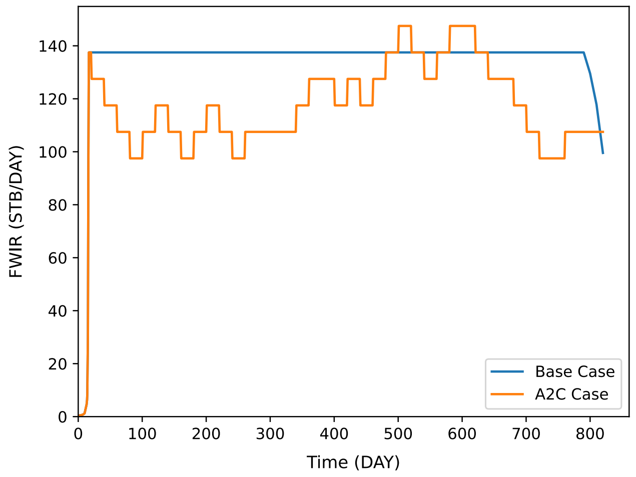
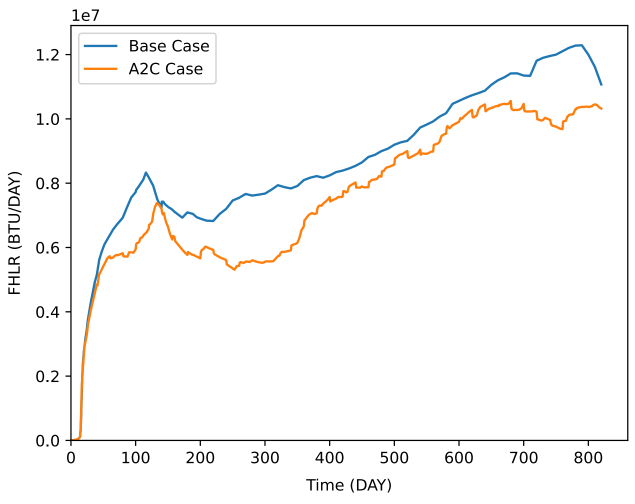
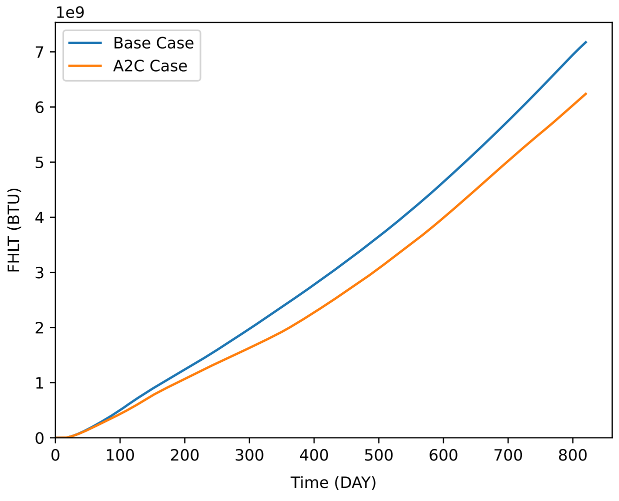


Fig. 6 Field water injection total versus time



(a) Heat losses per day versus time



(b) Cumulative heat losses versus time

Fig. 7 Field heat losses versus time

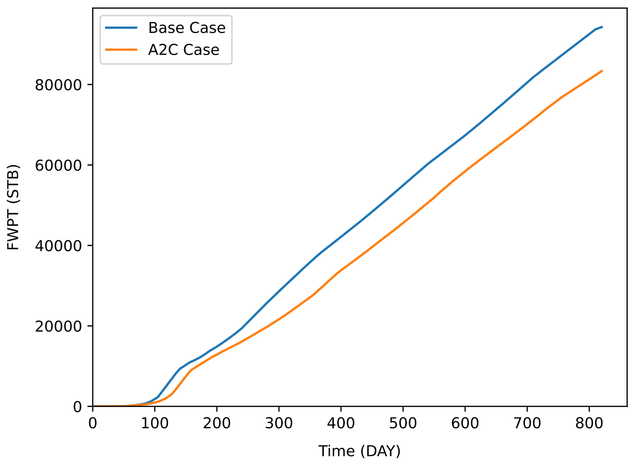


Fig. 9 Cumulative water production versus time

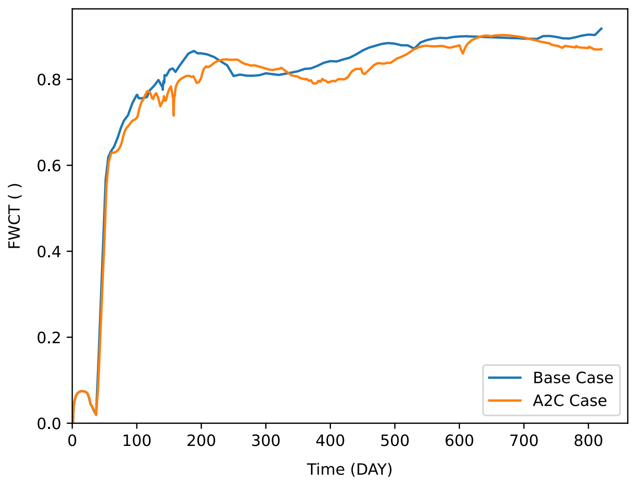
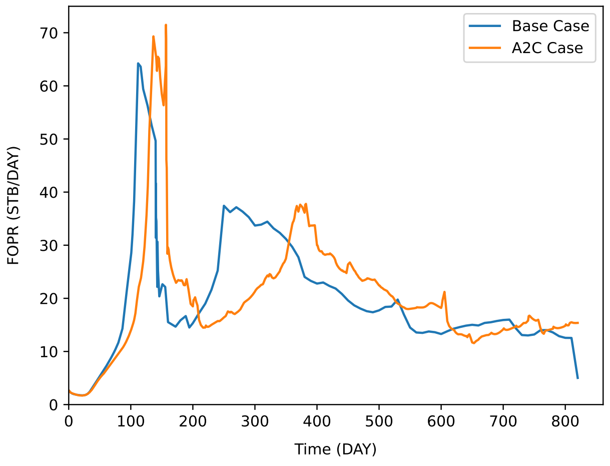
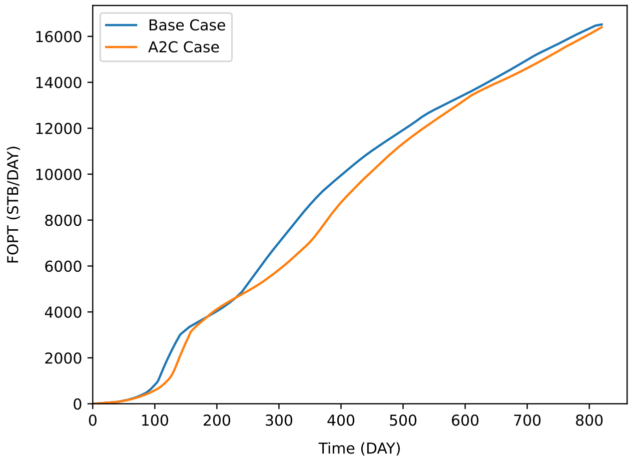


Fig. 8 Field water cut versus time

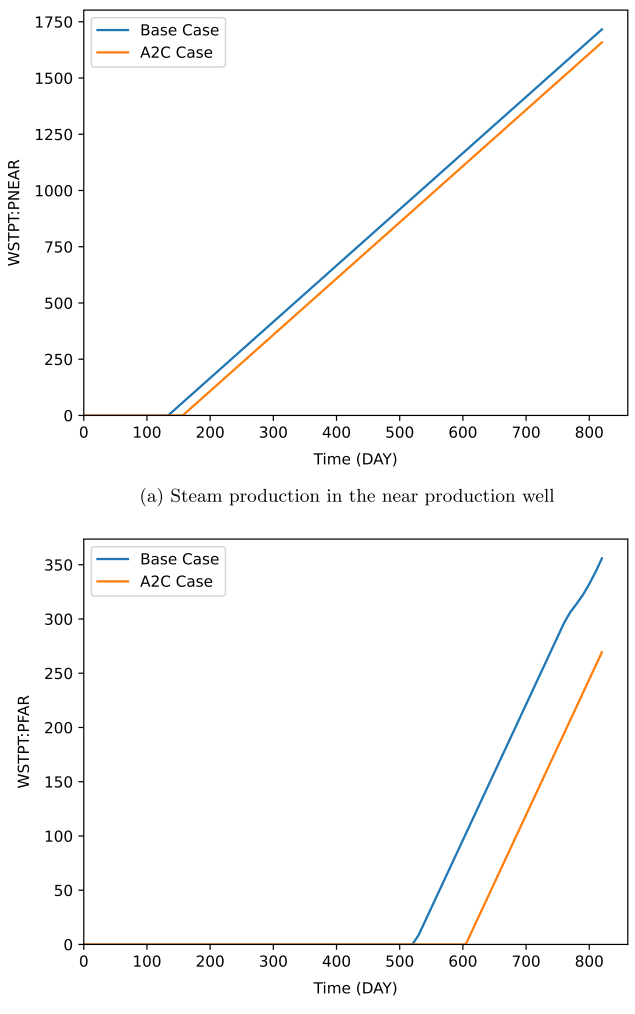


(a) Daily oil production rate



(b) Cumulative oil production

Fig. 10 Field oil production versus time



(a) Steam production in the near production well

(b) Steam production in the far production well

Fig. 11 Cumulative steam production per well versus time

cumulative loss to surrounding rocks. It is obvious that the policy defined by the (A2C) agent to change steam injection rate exhibits less heat loss to the surrounding rocks compared to the base case. This effect explains the success of this policy on water injection for optimizing oil recovery.

Also it is important to study the effect of this injection policy on the produced oil and water. From Figs. 8 and 9, it is shown that total water cut and the cumulative water production are less in the agent policy case than the base case. This may be an effect of slightly smaller injection rates at the first region of the production horizon that leads to the steam chamber to grew vertically.

Figure 10a, b shows the oil rate and cumulative oil production versus time. It is shown that the agent policy shifted the profile of oil production rate. The second peak shifted from day 260 to day 400. In the A2C policy this leads to reduce the sharp decrease happening afterward and also leads to delay the steam breakthrough in the near and far wells. Figure 11a, b shows the total steam production from the near and far wells in both A2C and base cases. While the near well starts to produce steam after 160 days in the A2C case while in the base case it starts to produce steam after 140 days. The same happens for the far well with approximately 80 days difference.

8 Conclusion

Steam injection is a common method to enhance the recovery from mature oil fields. Commonly, a constant steam rate is applied over a long period of time without considering varying physical phenomena and reservoir characteristics, consequently resulting in sub-optimal performance of these thermal heavy oil recovery processes. However, finding the optimal steam injection strategy is a challenge due to the complex dynamics, i.e., nonlinear formulation, variations over time, and reservoir heterogeneity. Such a problem can be reformulated using the Markov decision process for the application of reinforcement learning (RL). Subsequently, a decision-maker called an agent generates actions to maximize the production process's yield. The agent interacts continuously with the environment to determine the optimal injection rates, adjusting its actions based on feedback from the system until a satisfactory solution is reached.

In this work, an actor-critic RL architecture is used to train an agent to find the optimal strategy (i.e., policy) through continuously interacting with the environment. In this study, the environment is represented by a reservoir simulation model. At each time step, the agent executes an action to either increase, decrease, or keep the steam injection rate constant, and a subsequent reward is received by the agent. A reward can be defined as a distinct number from the environment. Then, the agent observes the new state from the environment. Such a state could be the cumulative amount of steam injected, the pressure distribution, or any other input that could be representative of the environment, which is the reservoir simulation model in our case. During this interaction, a policy function and a value function are trained. A policy gives a probability distribution of the actions that the agent can take. For a specific policy, a value function determines the expected yield for an agent starting from a given state. This dynamic process is executed for several episodes until convergence is achieved.

In this study, we propose an application of an actor-critical reinforcement learning (RL) approach to optimize steam injection rates based on a problem of compositional modeling of a fluid flowing through a porous medium. The objective was to find a policy that could maximize the cumulative net present value at the end of the production horizon by optimizing the well water injection rates. The purpose of the actor network is to estimate the value function through a mimic of the physics of the environment's inputs and outputs and to update the policy distribution in the direction suggested by the critic network. After a successive interaction between the agent and the reservoir simulation model, it is shown that the agent was able to choose a series of actions (steam injection rate versus time) aimed at maximizing the net present value of the project. The results from both the base and the optimum policy cases are compared from a physical perspective to explain the reasons leading to the increase in the net present value of the whole project.

This work represents an initial step in the direction of implementing a multi-agent reinforcement learning framework to optimize multi-pad oil wells.

Acknowledgements The authors wish to acknowledge financial support from Clausthal University of Technology through doctoral publications.

Funding Open Access funding enabled and organized by Projekt DEAL.

Data availability The reservoir model, e.g., "steam injection model" data that support the findings of this study are available in open porous medium repository, https://github.com/OPM/opm-data.

Declarations

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

References

1. Hou J, Zhou K, Zhang X-S, Kang X-D, Xie H (2015) A review of closed-loop reservoir management. Pet Sci 12(1):114–128. https://doi.org/10.1007/s12182-014-0005-6
2. Foss BA, Grimstad B, Gunnerud V (2015) Production optimization—facilitated by divide and conquer strategies. IFAC-PapersOnLine 48:1–8
3. Ali SMF, Meldau RF (1979) Current steamflood technology. J Pet Technol 31:1332–1342
4. Zhang J, Chen Z (2018) Formation damage by thermal methods applied to heavy oil reservoirs. In: Yuan B, Wood DA (eds) Formation damage during improved oil recovery. Gulf Professional Publishing, Houston, pp 361–384. https://doi.org/10.1016/B978-0-12-813782-6
5. Shafiei A, Dusseault MB (2013) Geomechanics of thermal viscous oil production in sandstones. J Pet Sci Eng 103:121–139
6. Ameli F, Mohammadi K (2018) A novel optimization technique for fast-SAGD process in a heterogeneous reservoir using discrete variables and repetition inhibitory algorithm. J Pet Sci Eng 171:982–992. https://doi.org/10.1016/j.petrol.2018.08.008
7. Saputelli L, Nikolaou M, Economides MJ (2005) Self-learning reservoir management. SPE Reserv Eval Eng 8(6):534–547. https://doi.org/10.2118/84064-PA
8. Patel K, Aske EMB, Fredriksen M (2014) Use of model-predictive control for automating SAGD well-pair operations: a simulation study. SPE Prod Oper 29:105–113
9. Purkayastha SN, Gates ID, Trifkovic M (2015) Model-predictive-control (MPC) of steam trap subcool in steam-assisted gravity drainage (SAGD). IFAC-PapersOnLine 48(8):539–544. https://doi.org/10.1016/j.ifacol.2015.09.023
10. Eaton AN, Beal LDR, Thorpe SD, Hubbell CB, Hedengren JD, Nybø R, Aghito M (2017) Real time model identification using multi-fidelity models in managed pressure drilling. Comput Chem Eng 97:76–84. https://doi.org/10.1016/j.compchemeng.2016.11.008
11. Vembadi SS, Patel RG, Prasad V (2018) Real-time feedback control of SAGD wells using model predictive control to optimize steam chamber development under uncertainty. Can J Chem Eng 96:1290–1305
12. Discrete-time MPC for Beginners (2009) In: Model predictive control system design and implementation using MATLAB®. Springer, London, pp 1–42. https://doi.org/10.1007/978-1-84882-331-0_1
13. Sibaweihi N, Patel RG, Guevara JL, Gates ID, Trivedi JJ (2021) Real-time steam allocation workflow using machine learning for digital heavy oil reservoirs. J Pet Sci Eng 199:108–168, ISSN 0920-4105. https://doi.org/10.1016/j.petrol.2020.108168.
14. Thévenin D, Janiga G (2008) Optimization and computational fluid dynamics. Springer, Berlin Heidelberg
15. Guevara JL, Patel R, Trivedi J (2021) Optimization of steam injection in SAGD using reinforcement learning. J Pet Sci Eng 206:108735. https://doi.org/10.1016/j.petrol.2021.108735
16. Mullapudi A, Lewis MJ, Gruden CL, Kerkez B (2020) Deep reinforcement learning for the real time control of stormwater systems. Adv Water Resour 140:103600. https://doi.org/10.1016/j.advwatres.2020.103600
17. Siraskar R (2021) Reinforcement learning for control of valves. Mach Learn Appl 4:100030. https://doi.org/10.1016/j.mlwa.2021.100030
18. Garnier P, Viquerat J, Rabault J, Larcher A, Kuhnle A, Hachem E (2021) A review on deep reinforcement learning for fluid mechanics. Comput Fluids 225:104973. https://doi.org/10.1016/j.compfluid.2021.104973
19. Abdalla R, El Ela MA, El-Banbi A (2020) Identification of downhole conditions in sucker rod pumped wells using deep neural networks and genetic algorithms (includes associated discussion). SPE Prod Oper 35(02):435–447. https://doi.org/10.2118/200494-PA
20. Abdalla R, Samara H, Perozo N, Carvajal CP, Jaeger P (2022) Machine learning approach for predictive maintenance of the electrical submersible pumps (ESPS). ACS Omega 7(21):17641–17651. https://doi.org/10.1021/acsomega.1c05881
21. Etesami D, Shirangi MG, Zhang WJ (2021) A semiempirical model for rate of penetration with application to an offshore gas field. SPE Drill Complet 36(01):29–46

22. Etesami D, Zhang WJ, Hadian M (2021) A formation-based approach for modeling of rate of penetration for an offshore gas field using artificial neural networks. J Nat Gas Sci Eng 95:104104. https://doi.org/10.1016/j.jngse.2021.104104
23. Manasipov R, Nikolaev D, Didenko D, Abdalla R, Stundner M (2023) Physics informed machine learning for production forecast. In: SPE reservoir characterisation and simulation conference and exhibition
24. Noshi CI, Eissa MR, Abdalla RM (2019) An intelligent data driven approach for production prediction. In: OTC offshore technology conference. https://doi.org/10.4043/29243-MS
25. Castelletti A, Pianosi F, Restelli M (2013) A multiobjective reinforcement learning approach to water resources systems operation: Pareto frontier approximation in a single run. Water Resour Res 49:3476–3486. https://doi.org/10.1002/wrcr.20295
26. Mahootchi M, Tizhoosh H, Ponnambalam K (2007) Reservoir operation optimization by reinforcement learning. J Water Manag Model. https://doi.org/10.14796/JWMM.R227-08
27. Gosavi A (1997) Simulation-based optimization: parametric optimization techniques and reinforcement. Learning. https://doi.org/10.1007/978-1-4757-3766-0
28. Abe T, Oh-hara S, Ukita Y (2021) Adoption of reinforcement learning for the intelligent control of a microfluidic peristaltic pump. Biomicrofluidics 15(3):034101. https://doi.org/10.1063/5.0032377
29. Brunton SL, Kutz JN (2019) Data-Driven science and engineering - machine learning, dynamical systems, and control. Cambridge University Press, Cambridge. https://doi.org/10.1017/9781108380690
30. Dworschak F, Dietze S, Wittmann M, Schleich B, Wartzack S (2022) Reinforcement learning for engineering design automation. Adv Eng Inform 52:101612. https://doi.org/10.1016/j.aei.2022.101612
31. de Lara FM, Ferrer E (2022) Accelerating high order discontinuous Galerkin solvers using neural networks: 1d burgers' equation. Comput Fluids 235:105274. https://doi.org/10.1016/j.compfluid.2021.105274
32. Marten-Guerrero JD, Lamata L (2021) Reinforcement learning and physics. Appl Sci. https://doi.org/10.3390/app11188589
33. Perera ATD, Kamalaruban P (2021) Applications of reinforcement learning in energy systems. Renew Sustain Energy Rev 137:110618. https://doi.org/10.1016/j.rser.2020.110618
34. Chen X, Yao L, McAuley J, Zhou G, Wang X (2023) Deep reinforcement learning in recommender systems: a survey and new perspectives. Knowl Based Syst 11:335. https://doi.org/10.1016/j.knosys.2023.110335
35. Liu H, Zhu D, Liu Y, Du A, Chen D, Ye Z (2018) A reinforcement learning based 3d guided drilling method: Beyond ground control. In: Proceedings of the 2018 VII international conference on network, communication and computing. ICNCC 2018. Association for Computing Machinery, pp 44–48. https://doi.org/10.1145/3301326.3301374. Accessed 27 May 2022
36. ArnØ M, Godhavn J-M, Aamo OM (2020) Deep reinforcement learning applied to managed pressure drilling. In: SPE, pp. 021–007001. https://doi.org/10.2118/200757-MS. https://onepetro.org/SPEBERG/proceedings/20BERG/2-20BERG/Virtual/448669 Accessed 27 May 2022
37. Yu Y, Chen W, Liu Q, Chau M, Vesselinov V, Meehan R (2021) Training an automated directional drilling agent with deep reinforcement learning in a simulated environment. In:SPE , pp. 041–013002. https://doi.org/10.2118/204105-MS. https://onepetro.org/SPEDC/proceedings/21DC/4-21DC/Virtual/460374 Accessed 27 May 2022
38. Ma H, Yu G, She Y, Gu Y (2019) Waterflooding optimization under geological uncertainties by using deep reinforcement learning algorithms, pp 031–043001. https://doi.org/10.2118/196190-MS. Accessed 18 Apr 2022
39. Hourfar F, Bidgoly HJ, Moshiri B, Salahshoor K, Elkamel A (2019) A reinforcement learning approach for waterflooding optimization in petroleum reservoirs. Eng Appl Artif Intell 77:98–116. https://doi.org/10.1016/j.engappai.2018.09.019
40. Miftakhov R, Al-Qasim A, Efremov I (2020) Deep reinforcement learning: reservoir optimization from pixels. In: IPTC. 10.2523/IPTC-20151-MS, pp 021–052002. . https://onepetro.org/IPTCONF/proceedings/20IPTC/2-20IPTC/Dhahran,%20Kingdom%20of%20Saudi%20Arabia/154747 Accessed 27 May 2022
41. Thuerey N, Holl P, Mueller M, Schnell P, Trost F, Um K (2021) Physics-based deep learning. arXiv version: 3. arXiv: 2109.05237. Accessed 24 July 2022
42. Sun A (2020) Optimal carbon storage reservoir management through deep reinforcement learning. Appl Energy. https://doi.org/10.1016/j.apenergy.2020.115660
43. Sun AY, Nicot J-P (2012) Inversion of pressure anomaly data for detecting leakage at geologic carbon sequestration sites. Adv Water Resour 44:20–29. https://doi.org/10.1016/j.advwatres.2012.04.006
44. Nguyen A-T, Reiter S, Rigo P (2014) A review on simulation-based optimization methods applied to building performance analysis. Appl Energy 113:1043–1058. https://doi.org/10.1016/j.apenergy.2013.08.061
45. De Paola G, Ibanez-Llano C, Rios J, Kollias G (2020) Reinforcement learning for field development policy optimization, pp. 041–046003. https://doi.org/10.2118/201254-MS. Accessed 18 Apr 2022
46. Dawar K (2021) Reinforcement learning for well location optimization. Pennsylvania State University, State College
47. Dong P, Chen Z-M, Liao X-W, Yu W (2022) A deep reinforcement learning (DRL) based approach for well-testing interpretation to evaluate reservoir parameters. Pet Sci 19(1):264–278. https://doi.org/10.1016/j.petsci.2021.09.046
48. Degris T, White M, Sutton RS (2012) Off-policy actor-critic. arXiv: 1205.4839
49. Bilgin E (2020) Mastering reinforcement learning with python: build next-generation, self-learning models using reinforcement learning techniques and best practices. Packt Publishing, Birmingham
50. Baird L (1995) Residual algorithms: reinforcement learning with function approximation. In: Prieditis A, Russell S (eds) Machine learning proceedings 1995. Morgan Kaufmann, San Francisco, pp 30–37. https://doi.org/10.1016/B978-1-55860-377-6.50013-X
51. Dong H, Ding Z, Zhang S (2020) Deep reinforcement learning—fundamentals, research and applications. Springer, Singapore
52. Sutton R, Barto A (2018) Reinforcement learning: an introduction. MIT Press, Cambridge. https://doi.org/10.1109/TNN.1998.712192
53. Garnier P, Viquerat J, Rabault J, Larcher A, Kuhnle A, Hachem E (2021) A review on deep reinforcement learning for fluid mechanics. Comput Fluids 225:104973. https://doi.org/10.1016/j.compfluid.2021.104973
54. Aziz K, Ramesh AB, Woo PT (1987) Fourth SPE comparative solution project: comparison of steam injection simulators. J Pet Technol 39(12):1576–1584. https://doi.org/10.2118/13510-PA