IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 2 May 2024, accepted 23 June 2024, date of publication 10 July 2024, date of current version 12 September 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3425837

RESEARCH ARTICLE

Efficient Scheduling of Constant Pressure Stratified Water Injection Flow Rate: A Deep Reinforcement Learning Method

JINZHAO HU[1], FUSHEN REN[1], ZHI WANG[2], AND DELI JIA[2]
[1]School of Mechanical Science and Engineering, Northeast Petroleum University, Daqing, Heilongjiang 163318, China
[2]Research Institute of Petroleum Exploration and Development, Beijing 100083, China

Corresponding author: Deli Jia (jiadeli422@petrochina.com.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 52074345, and in part by the Scientific Research and Technology Development Project of PetroChina under Grant 2021ZG12.

ABSTRACT Layered water injection is an important means of oilfield development. However, the process is fraught with challenges due to the complexity of the subsurface environment and the variability in water absorption properties across different strata. The water injection flow rate of each layer is affected by various factors, resulting in a typical incomplete system and extremely complex measurement and adjustment of layered water injection. The reasons include development plans, water absorption properties of each layer, and differences in the structure of underground water injection devices. The combined effect of these factors leads to the typical incomplete system and complexity of the measurement and regulation of the layer section of layered water injection. In this paper, a reinforcement learning-based stratified water injection control algorithm was proposed to solve the problem of stratum flow scheduling in the complex environment of wellbore during stratified water injection. The reinforcement learning algorithm is combined with the differential pressure stratified water injection control algorithm, and a reasonable reward function is set according to the injection error to improve the algorithm control accuracy and efficiency. In order to evaluate the performance of the algorithm, a gym-based simulation environment is established to simulate the nonlinear stratigraphic environment under stochastic conditions, so that the model has a better generalization performance. Compared with other algorithms, the proposed stratified water injection control algorithm saves 41.94% of training time, the average injection error is less than 5% in the water injection environment with different number of stratum segments, the average success rate is more than 90%, and there is 85% probability to reach the injection target within 1-5 steps, which provides a more excellent performance in terms of control accuracy and adjustment speed. The algorithm has an important guiding role in the flow scheduling control of injection wells and realizing the automation of layered water injection process. Our code will be available online at: https://github.com/HJZ-hub/ASWI.

INDEX TERMS Deep reinforcement learning, layered water injection, SAC, flow scheduling.

I. INTRODUCTION

Water injection development can effectively maintain interlayer pressure, regulate interlayer contradictions, and increase the fluctuation coefficient of water injection, which is an effective method to ensure high and stable production of oil fields [1], [2], [3]. The water injection system is subject to the interaction and influence of many factors, such as water injection equipment, pipeline parameters, formation environment and geo-engineering, etc. The water injection environment is complex and variable [4]. On the one hand, the water injection process is influenced by the friction

The associate editor coordinating the review of this manuscript and approving it for publication was Xinyu Du.

of the pipeline, the compressibility of the liquid and the elasticity of the pipeline, especially in the localized resistance part of the pipeline, this nonlinear characteristic is especially obvious [5]. On the other hand, factors such as permeability, porosity and fluid saturation of the formation are nonlinear and vary with the injection pressure and volume of water injection [6]. This further adds to the complexity and nonlinearity of the water injection environment. The fourth generation cable controlled layered water injection technology, with full process digitization, automation, and tool integration, achieves underground data transmission and layer flow control through cables, greatly improving the efficiency of water injection well testing and adjustment [7], [8]. However, due to the complex underground environment, long interval control time, and high control frequency, the service life of the water injection device is seriously affected. The number of subdivided layer sections is large, and the water injection precision is low, which can not ensure good water injection efficiency. With the promotion and application of intelligent oilfield has higher requirements for water injection effect [9]. Unattended automatic deployment technology has become the difficulty and focus of the intelligent development of oilfield wellbore.

In recent years, many scholars have conducted in-depth research on flow control in injection wells using finite element analysis and experimental methods. Among them, Zhou et al. [10] used numerical calculation methods to calculate the injection pressure by combining the flow and injection index curve in the pipe column, and used the nozzle flow equation to calculate the nozzle size using the nozzle front pressure and injection pressure. Jiang et al. [11] established a method for optimizing the layered water injection nozzle in heterogeneous reservoirs by studying the relationship between the layered water injection nozzle of water injection wells, the injection volume and pressure of each sub layer, and the physical properties of each sub layer. Xiufang's indoor experimental method was to simulate the flow injection situation under different wellhead pressures and water nozzle openings indoors, and the experiment verified that the injection flow rate is directly proportional to the 0.5 power of the pressure difference. The numerical simulation method involves establishing a geometric model of a water nozzle and conducting numerical simulations. Wang et al. [12] investigated the pressure loss of fluids with different densities and viscosities in nozzles, and ultimately determined the relationship between nozzle pressure loss and flow rate, nozzle inner diameter, and fluid density. At present, the injection flow rate of water injection wells is calculated using the nozzle flow equation method, but there are differences in the structural parameters of the research objects, and linear equations are difficult to solve nonlinear problems in the water injection process [13]. Reinforcement learning algorithms have advantages in dealing with decision control problems in nonlinear environments. Through continuous experimentation and feedback, they can gradually learn the optimal decision strategy, adapt to changing environments and parameter differences, and optimize system performance to the greatest extent possible. Therefore, we can try to solve this problem [14], [15].

Reinforcement learning is a machine learning technique for solving optimal decision problems in dynamic environments [16], [17]. It has shown performance equivalent to and beyond human levels in fields such as robot motion control [18], [19], path planning [20] and resource scheduling [21], [22], [23]. Numerous scholars have applied reinforcement learning techniques to the control of robots in complex environments as well as to the problem of resource allocation, and are committed to solving the environmental complexity and nonlinearities faced during the control of intelligent bodies [24], [25], [26]. Iklassov et al. [20] proposed a novel end-to-end framework for solving the Vehicle Route Problem with Stochastic Demand (VRPSD) using a reinforcement learning approach, which outperforms previous state-of-art metaheuristic algorithms in terms of performance and exhibits better robustness to environmental changes. Qi et al. [21] proposed a UAV control strategy based on a deep deterministic policy gradient, which solves the problem of multi-UAV three-dimensional movement and energy replenishment scheduling, and realises efficient and fair coverage for users in large areas. Si et al. [23] described the decision-making process of home energy by building a mathematical model and used algorithms such as deep Q-learning and deep deterministic policy gradient for energy management. Perrusquía et al. [25] proposed a multi-intelligence reinforcement learning based approach to solve the kinematic problem of redundant robots and provided a task-space control scheme applicable to multi-linked robots. The study of reinforcement learning in dealing with control and resource allocation problems in complex environments provides important lessons for flow scheduling control during stratified water injection.

In the process of stratified water injection, the flow rate of the current stratum section is not only related to the current injection environment but also related to the action of the current water injection device. As shown in Figure 1 in the water injection process, the water injection device observable parameters include wellhead pressure, formation pressure, formation depth, nozzle opening, wellhead flow rate, the water injection device as an intelligent body will be regulated as the time of the intelligent body action, so that the water

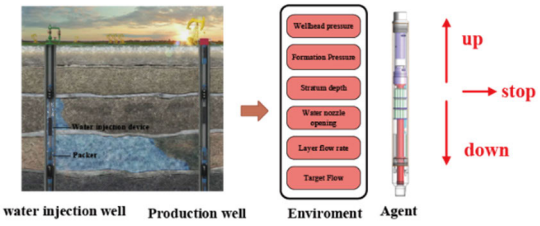


FIGURE 1. Layered water injection device control and working principle.

injection device to complete the upward adjustment, downward adjustment, and closure of the control action.

In this paper, a SAC-based reinforcement learning method for stratified water injection is proposed for solving the injection flow scheduling problem during stratified water injection, and the main contributions can be listed as:

First, a nonlinear simulation environment based on the water injection law, which can simulate the water injection process in any layer section with any formation pressure under the constant pressure condition at the wellhead, is established for training and validating the algorithm proposed in this paper.

Second, a method for calculating the water nozzle of stratified water injection under constant pressure is proposed, and the model can more accurately calculate the water nozzle opening of the stratified water injection device under this pressure.

Third, a SAC-based hierarchical water injection reinforcement learning method is proposed, which is compared with the SAC algorithm and the PFC (Baseline) hierarchical water injection computational model, and the algorithm is able to better achieve the water injection goal of the water injection device in a nonlinear environment.

II. CALCULATION METHOD FOR THE OPENING OF DOWNHOLE WATER INJECTION DEVICES

A. HYDRODYNAMIC MODELING OF WATER INJECTION COLUMNS

In the process of water injection, the effect of water injection is jointly determined by parameters such as surface equipment, downhole tools, formation water absorption capacity, and water nozzle shape. As shown in Figure 2 for the layered water injection structure, this model is based on the actual injection process, based on the observable parameters according to the wellhead pressure P_0, stratum section formation pressure P_i, depth of the bottom layer of the water injection device h_i, percentage of nozzle opening x_i, and injection flow rate of the stratum section q_i, and the fluid dynamics in the water injection column is modeled.

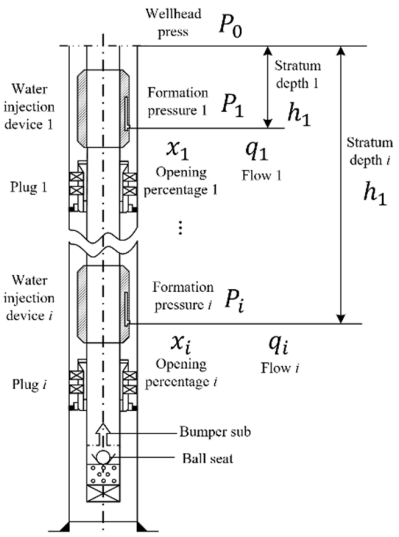


FIGURE 2. Water injection well structure.

The water injection process is regarded as incompressible flow, when the water nozzle is no longer changed, the velocity and pressure of the fluid will not change with the change of time, the total energy per unit mass of fluid is kept constant in any two flow lines, and the hydraulic equilibrium relationship between the wellhead and the outlet of the nozzle of each layer section can be obtained from Bernoulli's equation [29]:

$$\frac{P_0}{\rho g} + h_i = \frac{P_i}{\rho g} + \frac{v_i^2}{2g} + h_{wi} \tag{1}$$

where, P_0 is the wellhead pressure, Pa; ρ is the density of the injected liquid, kg/m^3; g is the acceleration of gravity, m/s^2; P_i is the formation pressure of layer i, Pa; v_i is the average velocity of the liquid at the outlet of the nozzle of layer i, m/s; h_i is the depth of the formation from the wellhead to the layer i, m; and h_{wi} is the head loss from the wellhead to the layer i, m.

The head loss includes along-travel loss and localized loss, and its expression is [28]:

$$h_{wi} = h_{fi} + h_{ji} \tag{2}$$

where, h_{fi} denotes the along-track loss from the wellhead to layer i, m; and h_{ji} denotes the localized loss at the exit of layer i, m.

A change in pipe diameter at the outlet of the water injection unit produces a localized pressure loss expression [29]:

$$h_{ji} = \zeta_i \frac{v_i^2}{2g} \tag{3}$$

where, ζ_i is the local loss coefficient of the layer i nozzle, generally obtained by experiment; v_i is the average velocity of the liquid at the outlet of the layer i nozzle, m/s.

The along-travel resistance loss is the energy loss resulting from relative motion within the liquid and viscous friction between the liquid and the pipe wall, expressed as:

$$h_{fi} = \begin{cases} \lambda_i \dfrac{h_i}{d_o} \dfrac{v_{oi}^2}{2g}, & i = 1 \\ \lambda_1 \dfrac{(h_1)}{d_o} \dfrac{v_{o1}^2}{2g} + \displaystyle\sum_{i=2}^{n} \lambda_i \frac{(h_i - h_{i-1})}{d_o} \frac{v_{oi}^2}{2g}, & i > 1 \end{cases} \tag{4}$$

where, λ_i is the loss coefficient in the layer i of the injection column, d_o is the diameter of the injection column, m; v_{oi} is the average velocity inside the injection column of the layer i, m/s.

The flow condition of the fluid, which has a large influence on the along-stream loss, is usually described by the Reynolds number, which is expressed as [28] for the ith section:

$$Re_i = \frac{\rho v_{oi} d_o}{\mu} = \frac{v_{oi} d_o}{\nu} \tag{5}$$

where, ρ is the fluid density, kg/m^3; v_{oi} is the average velocity in the layer i of the injection column, m/s, μ is

the hydrodynamic viscosity, Pa· s; v is the fluid kinematic viscosity, m^2/s; and d_o is the diameter of the injection column, m.

The pipe flow is transformed from laminar flow to transitional flow and turbulent flow, and the along-travel loss coefficient λ of the pipe exists a certain relationship with the Reynolds number and the relative roughness of the pipe in different flow states, and the expression for the relative roughness of the water injection pipe column is:

$$\overline{\varepsilon} = \frac{\varepsilon}{d_o} \tag{6}$$

where, the relative roughness $\overline{\varepsilon}$ is a dimensionless number, ε is the roughness of the inner surface of the pipe, m; and d_o is the diameter of the water injection pipe column, m.

When the Reynolds number $Re < 2300$, there is a laminar flow in the injection column and the along-stream resistance coefficient is expressed as:

$$\lambda_i = \frac{64}{Re_i} \tag{7}$$

When the Reynolds number Re > 2300, the injection column is turbulent flow, and the along-stream resistance coefficient is expressed by using Haaland formula [10] as:

$$\frac{1}{\sqrt{\lambda_i}} = -1.8\lg\left[\left(\frac{\overline{\varepsilon}}{3.7}\right)^{1.11} + \frac{6.9}{Re_i}\right] \tag{8}$$

The method of calculating the loss factor in the layer i of the injection column can be expressed as follows: [27]:

$$\lambda_{oi} = \begin{cases} \dfrac{64}{Re_i}, & Re < 2300 \\ \left\{-1.8\lg\left[\left(\dfrac{\overline{\varepsilon}}{3.7}\right)^{1.11} + \dfrac{6.9}{Re_i}\right]\right\}^{-2}, & Re \geq 2300 \end{cases} \tag{9}$$

B. MODELING OF CONSTANT PRESSURE WATER INJECTION SIMULATION ENVIRONMENT

As shown in Figure 3 U-type water nozzle structure, layered water injection device nozzle throttling hole, need to meet the gradient of the overflow area is easy to adjust, hydraulic radius, resistance to blockage, flow control range, etc., the shape of the water nozzle directly determines the nozzle on the flow of the flow regulation effect is good or bad, is an important part of the water injection device. The expression

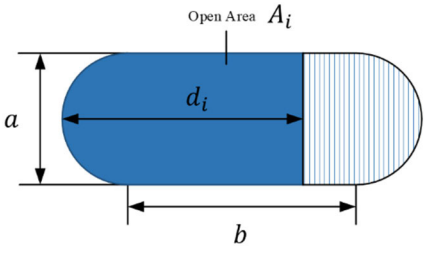


FIGURE 3. U-shaped nozzle structure.

for calculating the overflow area A_i of the water injection device in the layer i of the injection column is as follows:

$$A_i(x_i, a, b) = \begin{cases} \dfrac{a^2}{4}\arccos\left(1 - \dfrac{2d_i}{a}\right) - \left(\dfrac{a}{2} - d_i\right)\sqrt{ad_i - d_i^2}, \\ 0 \leq d_i < \dfrac{a}{2} \\ \dfrac{\pi a^2}{8} + a\left(x_i - \dfrac{a}{2}\right), \\ \dfrac{a}{2} \leq d_i \leq b + \dfrac{a}{2} \\ \dfrac{\pi a^2}{4} + ab - \dfrac{a^2}{4}\arccos\left(\dfrac{2(d_i - b)}{a} - 1\right) \\ + \left(d_i - \dfrac{a}{2} - b\right)\sqrt{2d_i b + d_i a - d_i^2 - b^2 - ab}, \\ b + \dfrac{a}{2} < d_i \leq b + a \end{cases} \tag{10}$$

where, a represents the height of the U-shaped nozzle, m; b represents the width of the U-shaped nozzle, m; d_i represents the opening of the nozzle, m.

The fluid velocity v_{oi} in the injection column is expressed by the fluid continuity equation as:

$$v_{oi} = \frac{v_i A_i}{A_o} = \frac{4v_i A_i}{\pi d_o^2} \tag{11}$$

where, v_i denotes the average velocity of the liquid at the outlet of the nozzle of the layer i, m/s; A_i denotes the overflow area of the nozzle, m^2; A_o denotes the cross-sectional area of the water injection pipe column, m^2; d_o denotes the diameter of the water injection pipe column, m.

Bringing Equation (11) into Equation (4) yields a functional relationship between the outlet flow rate v_i at the nozzle and the along-stream loss, and bringing Equation (4) and Equation (3) into Equation (1) yields an expression for the outlet flow rate:

$$v_i(P_0, h_i, P_i, \zeta_i, \lambda_{oi}) = \sqrt{\frac{2P_0 + 2\rho g h_i - 2P_i}{\rho + \zeta_i\rho + \lambda_{oi}\frac{h_i}{d_o}\frac{A_i^2}{A_0^2}\rho}} \tag{12}$$

The nozzle outlet flow rate vi, brought into Equation 13 can be calculated to obtain the expression for the water injection in layer i as:

$$Q_i = A_i v_i \tag{13}$$

The loss coefficient λ_{oi} along the injection column is a function of the velocity v_{oi} inside the injection column, which cannot be obtained when the nozzle outlet flow rate is unknown. Using the iterative method, assuming that the coefficient of loss along the injection column $\lambda'_{oi} = 0.01$, bring into equation (12) to get the nozzle outlet flow velocity v_i under the opening, bring into equation (11) to get the average velocity v_{oi} in the injection column, calculate the Reynolds number Re_i under the flow velocity, and then bring into equation (9) to get the coefficient of loss along the nozzle outlet flow velocity λ_{oi}, which can be calculated by using the original input coefficient of loss along the injection column

Algorithm 1 An Iterative Method for Calculating the Flow Rate of Layer Segments in a Constant-Pressure Environment

Initialization:
Nozzle dimensions a, b; number of layer segments n, along-track loss coefficient error γ; wellhead pressure P_0; formation pressure $P = [P_1, P_2 \ldots P_i]$, height of each layer segment from the surface $h \leftarrow [h_1, h_2, \ldots h_i]$; spool position of each layer segment $d \leftarrow [d_1, d_2, \ldots d_n]$; empty flow output set $Q = [\,]$; $i \in [1, 2, \ldots n]$.

for $i \leftarrow 1$ **to** n **do**
  Bring the spout opening x_i, U-shaped spout dimensions a, b into Equation (10) to calculate the spout area A_i.
  Determine the local loss coefficient ζ_i for the spigot.
  $\lambda_i' = 0.01$.
  repeat
    $\lambda_i \leftarrow \lambda_i'$ is brought into Equation. (11) to calculate the nozzle outlet flow rate v_i.
    Bring v_i into Equation (12) to find v_{oi}.
    Calculate the Reynolds number Re and the relative roughness $\overline{\varepsilon}$
    Equation (9) finds the current front range loss coefficient λ_i'.
  until convergence ($|\lambda_i - \lambda_i'| \leq \gamma$)
  λ_i is brought into Equation (12)(13) to find the layer i flow Q_i
  Add Q_i to the matrix $Q = [\,]$.
end for

λ'_{oi}, with the original input loss coefficient v_{oi}. The coefficient of along-range loss λ_{oi} is compared with the original input along-range loss coefficient λ'_{oi}, and the approximate nozzle exit velocity v_i is obtained through continuous iteration so as to calculate the water injection volume of each layer segment, and the calculation process is shown in Algorithm 1:

Aiming at the nonlinear phenomenon existing in the water injection process, the constant pressure water injection environment with different number of layer segments is constructed based on gym. The local loss coefficient ζ is an important parameter affecting the water injection volume in the process of water injection, and the random error N is added to the local loss coefficient ζ to make the result of the water injection volume have nonlinearity, and the expression of the local loss coefficient ζ_i^* after the addition of noise is as follows:

$$\zeta_i^* = \zeta_i + N \tag{14}$$

where, ζ_i is the localized loss coefficient of the i-layer and the noise N obeys the normal distribution $N \sim N(0, 0.158)$.

As shown in Figure 4, the simulation results of water injection in the layer section with different nozzle pressure differences under 50% nozzle opening are shown, from which it can be seen that this injection environment has nonlinear characteristics.

C. CALCULATION OF THE OPENING OF THE NOZZLE OF THE WATER INJECTION DEVICE

Based on the single-layer segment flow rule of change proposed in the literature [10], [11], [12], this paper proposes

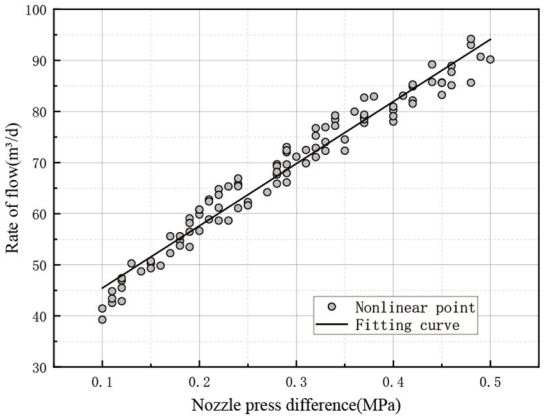


FIGURE 4. Nonlinear differential pressure flow chart.

a constant-pressure environment of the water nozzle flow control method, the detailed step calculation process is as follows.

With a known target injection flow rate for the layer section, the flow velocity v_{oi} in the injection column is expressed as:

$$v_{oi} = \frac{Q_{tai}}{d_o} \tag{15}$$

where, Q_{tai} is the target injection volume in layer i, m/s; and d_o is the diameter of the water injection column, m.

The pre-nozzle pressure is the pressure measured inside the column of tubing at the outlet of the injection device, which is jointly determined by the wellhead pressure, depth of the injection layer, and head loss, and the expression for the pre-nozzle pressure is:

$$P_{bi} = P_o + \rho g h_i - h_{fi} \tag{16}$$

The injection differential pressure is the difference between the pressure in front of the injection device pre-nozzle and the formation pressure, expressed as:

$$\Delta P_i = P_{bi} - P_i \tag{17}$$

A large number of studies have shown that there is a linear relationship between injection differential pressure and flow rate and nozzle opening, and the expression between nozzle opening and flow rate can be obtained through experimental or simulation methods [10], [11], [12]:

$$d_i = \left(\frac{Q_{tai} + C}{K \Delta P_i^{0.5}} \right)^N \tag{18}$$

where, d_i denotes the spout opening, m; C, K, and N are constants related to the shape of the spout, which are generally obtained by experiment or simulation. Experimentally shown that $K = 14.07$, $C = 3.14$, $N = 1.01$.

The formula (14) into the formula (5) can get the layer section of the Reynolds number Re, the Reynolds number into the formula (9) can be obtained along the loss coefficient λ_i, along the loss coefficient into the formula (4) can be obtained along the loss of the section of the loss of h_fi, which

Algorithm 2 Pressure Flow Control algorithm(Baseline)

Initialization:

Nozzle dimensions a, b; water injection column diameter d_o; formation pressure $P = [P_1, P_2, \ldots, P_i]$; injection device depths $h = [h_1, h_2, \ldots, h_i]$; segment target injection volume $Q_{ta} = [Q_{ta1}, Q_{ta2}, \ldots, Q_{tai}]$; $i \in [1, 2, \ldots, n]$.

Nozzle dimensions a, b; injection column diameter d_o; formation pressure $P = [P_1, P_2, \ldots, P_i]$; depth of the injection device $h = [h_1, h_2, \ldots, h_i]$; stratum segment target injection volume $Q_{ta} = [Q_{ta1}, Q_{ta2}, .., Q_{tai}]$; $i \in [1, 2, \ldots n]$.

1: **for** $i \leftarrow 1$ **to** n **do**

2: Equation (14) calculates the fluid velocity v_{oi} in this section of the column.

3: Calculate the Reynolds number of the pipe column in this layer section $Re = \frac{v_{oi} d_o}{\nu}$.

4: Equation (4) calculates the along-travel loss coefficient h_{fi} for this section of the pipe column.

5: Equation (16) yields the nozzle differential pressure ΔP_i

6: Water injection device opening, $d_i = \left(\frac{Q_{tai} + C}{K \Delta P_i^{0.5}} \right)^N$.

7: Converted to percentage $x_i = d_i/(a + b)$.

8: **return** Set of water injection device openings for each layer segment $x = [x_1, x_2 \ldots x_i]$.

can be calculated through the formula (15) and formula (16) to get the differential pressure of the nozzle, and into the formula (17) to calculate the layer section of the opening of the injection device, and then the pressure flow is calculated through the formula (15) and formula (16), which can be calculated through the formula (17). Pressure Flow Control algorithm flow is shown in Algorithm 2.

III. DESIGN AND IMPLEMENTATION OF REINFORCEMENT LEARNING ALGORITHM FOR LAYERED WATER INJECTION

A. MARKOV DECISION PROCESS

Markov Decision Process (MDP) is a classical decision process in reinforcement learning [30]. The conditional probability of an intelligent future state depends not only on the current state, but also on the action taken by the intelligent in the current state [31], [32]. The process of regulating the flow rate of a stratified water injection layer segment can be regarded as a Markov decision process, where the flow rate of the current layer segment is not only related to the current dispensing environment but also to the current action of the water injection device.

A Markov process (MP) is a stochastic process whose conditional probability distribution of future states, given the present state and all past states, depends only on the present state [16]. This means that given the present state, it is conditionally independent of the past states Describe a Markov process with the tuple $< \mathcal{S}, \mathcal{P} >, \mathcal{S} = \{s_1, s_2, \ldots s_n\}$ denoting a finite number of sets of states, and P is the state transfer matrix with the expression:

$$\mathcal{P} = \begin{bmatrix} P(s_1 \mid s_1) & \cdots & P(s_n \mid s_1) \\ \vdots & \ddots & \vdots \\ P(s_1 \mid s_n) & \cdots & P(s_n \mid s_n) \end{bmatrix} \tag{19}$$

Row i and column j in $\mathcal{P}$ denote the probability of transforming from state $\boldsymbol{s}_i$ to $\boldsymbol{s}_j$, which is satisfied by the Markov process:

$$P(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t) = P(\boldsymbol{s}_{t+1} \mid \boldsymbol{h}_t) \tag{20}$$

where, $h_t = \{\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots \boldsymbol{s}_t\}$ denotes the history of the state, and the transfer from the current state $\boldsymbol{s}_t$ to $\boldsymbol{s}_{t+1}$ is equal to the transfer of all previous states to $\boldsymbol{s}_{t+1}$.

The reward function G_t is added to the Markov process, which is transformed into Markov Reward Process (MRP).The Markov Reward Process consists of the tuple $< \mathcal{S}, \mathcal{P}, r, \gamma >$, $r(\boldsymbol{s})$ is the reward function, which is the expectation of the reward that can be obtained when transferring to a certain state $\boldsymbol{s}$. γ denotes the discount factor, as shown in Figure 5 for the Markov Reward Process.

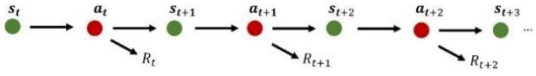


FIGURE 5. Markov reward process.

The sum of all reward decays from moment t, $\boldsymbol{S_t}$ up to the termination state is called the reward G_t(Return) and is denoted as:

$$G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma R_{t+k} \tag{21}$$

where, R_t denotes the reward obtained at moment t, and $\gamma \in [0, 1]$ denotes the discount factor, which responds to the effect of future rewards on the current harvest.

By adding action $\boldsymbol{a}$ to the original Markov reward process, a Markov decision process (MDP) is obtained consisting of tuples $\langle \mathcal{S}, \mathcal{A}, P, r, \gamma \rangle$.

Where S is the set of states, A denotes the set of actions, $r(\boldsymbol{s}, \boldsymbol{a})$ denotes the reward obtained by choosing action $\boldsymbol{a}$ in state $\boldsymbol{s}$, and $P(\boldsymbol{s}' | \boldsymbol{s}, \boldsymbol{a})$ denotes the probability that the state of performing action $\boldsymbol{a}$ in state $\boldsymbol{s}$ will change from the state of $\boldsymbol{s}$ to $\boldsymbol{s}'$, which is satisfied by the Markov decision process [30]:

$$P(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, \boldsymbol{a}) = P(\boldsymbol{s}_{t+1} | \boldsymbol{h}_t, \boldsymbol{a}_t) \tag{22}$$

The Markov decision process is a time-dependent uninterrupted process with non-stop interactions between the intelligence and the environment MDP. The strategy of the intelligence, denoted by π, is the probability of a in case the input state is $\boldsymbol{s}$ that is:

$$\pi(\boldsymbol{a} \mid \boldsymbol{s}) = P(\boldsymbol{A}_t = \boldsymbol{a} \mid \boldsymbol{S_t} = \boldsymbol{s}) \tag{23}$$

$V_\pi(\boldsymbol{s})$ denotes the state-value function based on policy π in MDP, representing the return obtained from state $\boldsymbol{s}$ according to policy π. The value of state $\boldsymbol{s}$ equals the sum of the products of probabilities and corresponding values for all actions taken under policy π. The expression is [35]:

$$V^\pi(\boldsymbol{s}) = \sum_{\boldsymbol{a} \in \mathcal{A}} \pi(\boldsymbol{a} \mid \boldsymbol{s}) Q^\pi(\boldsymbol{s}, \boldsymbol{a}) \tag{24}$$

$Q^{\pi}(\boldsymbol{s}, \boldsymbol{a})$ denotes the action-value function based on the policy π in the MDP, which represents the expected reward obtained by executing action $\boldsymbol{a}$ in state $\boldsymbol{s}$. It is equal to the product of the immediate reward plus the probability of all possible state transfers after decay and the corresponding value, with the expression [36]:

$$Q^{\pi}(\boldsymbol{s}, \boldsymbol{a}) = \gamma \sum_{s' \in \mathcal{S}} P\left(s' \mid \boldsymbol{s}, \boldsymbol{a}\right) V^{\pi}\left(s'\right) \tag{25}$$

B. REWARD FUNCTION

In the design of the water injection algorithm, the intelligent body (water injection device) contains two objectives, the smallest possible injection error, and the smallest possible number of adjustments in each layer segment.

The target injection volume of the stratum segment is q_{ta}, the actual injection volume is q_r, and the injection well has n stratum segments, and the accumulated total error $Ea(q_{ta}, q_r)$ is:

$$Ea_i(q_a, q_r) = \sum_{i=1}^{n} \left| q_{a,i} - q_{r,i} \right| \tag{26}$$

where, $q_{ta} \in \{q_{ta1}, q_{ta2}, \ldots, q_{ai}\}$ denotes the set of layer segment target injections, m^3/d, and $q_r \in \{q_{r1}, q_{r2}, \ldots, q_{ri}\}$ denotes the set of actual injections in the layer segments, m^3/d. The goal of the incentives is to adjust the layer segment injections in the direction of error reduction.

Motion reward R_{runi} is the control reward for the nozzle motion direction, the nozzle opening is proportional to the flow rate under constant pressure state, and the nozzle motion direction reward is given based on the target injection volume and the actual injection volume. The motion reward is denoted as:

$$R_{runi} = \begin{cases} R_{positive}, & (q_{tai} - q_{ri}) > 0 and ro_i > 0 \\ R_{reversal}, & (q_{tai} - q_{ri}) < 0 and ro_i < 0 \end{cases} \tag{27}$$

where, ro_i denotes the time and direction of rotation of the layer i segment, when $ro_i > 0$ means rotating ro_i seconds in the direction of increasing spout opening. Conversely, rotate ro_i seconds in the opposite direction.

The positional reward R_{loci} is to prevent the water nozzle movement to the critical value to prevent the maximum or minimum situation, the opening is zero when the water injection device layer section injection is zero and seriously affect the life of the water injection device, in order to prevent this situation, agent penalty is given when $x_i = 0$ or $x_i = 100$. The position reward is denoted as:

$$R_{loci} = -k \tag{28}$$

The purpose of the time reward R_{time} is to expect the agent to reach the goal in as few adjustment steps as possible, different number of layer segments have different time rewards, *levelecount* indicates the number of layer segments. The time reward incentive denoted as:

$$R_{time} = -0.5 * levelcount \tag{29}$$

The error reward R_{erroi} indicates the distance between the actual value and the target value, the continuous reward can be a good response, the distance between the agent and the target value, the error indicates the distance between the actual value and the target value. The expression of the error is:

$$Ea_i(q_a, q_r) = \sum_{i=1}^{n} |q_{ai} - q_{ri}| \tag{30}$$

Ten times the percentage of the error to the target value is used as the penalty value for the error reward, which is denoted as:

$$R_{erroi} = 10 \times \frac{\left| q_{tai} - q_{ri} \right|}{q_{tai}} \tag{31}$$

The target reward R_{target} is the reward when the agent reaches the error range of the target value, and the target reward is denoted as:

$$R_{target} = \begin{cases} R_{reach}, & success \\ R_{collision}, & collosion \end{cases} \tag{32}$$

Then the total reward function is expressed as:

$$R = \begin{cases} R_{target}, & finish \\ R_{time} + \sum_{i=1}^{n} R_{runi} + R_{loci} + R_{erroi}, & others \end{cases} \tag{33}$$

The reward values have been adjusted after extensive experiments, and the results show that the inclusion of the target reward can make the whole system converge more rapidly. The reward values for each part are, $R_{positive} = 1$, $R_{reversal,} = 1$, $k = 100$, $R_{reach} = 0$ and $R_{collision} = 100$.

C. STATE SPACE

Reinforcement learning algorithms make decisions and evaluate gains based on the state of the environment perceived by the robot, and a good state space design directly affects the efficiency and effectiveness of the algorithm [37].

The environmental state information includes ground information, layered injection unit structure information, real-time sensor information, formation information, and target injection volume information, and the state space is defined as:

$$S = \{p_0, x_i, h_i, p_i, ta_i\} \forall i \in (1, 2 \cdots, n) \tag{34}$$

where, p_0 denotes the wellhead injection pressure; x_i denotes the percentage of nozzle opening in layer i; h_i denotes the depth of the water dispenser in layer i; p_i denotes the formation pressure in layer i; and ta_i denotes the target injection volume in layer i.

D. ACTION SPACE

Layered water injection device is by controlling the size of the nozzle opening to change the nozzle differential pressure so as to realize the requirements of achieving the injection flow rate of different layers, here the nozzle movement time ro_i is

used as a parameter of the agent continuous action space [38], and the continuous action space A is defined as:

$$A = \{ro_i\} \forall i \in (0, 1, 2 \cdots, n) \tag{35}$$

where, $ro_i \in [-20, 20]$ unit seconds, when $ro_i > 0$ that means the time when the spool in the water distributor moves ro_i seconds in the direction of increasing the overflow area, $ro_i = 0$ that means the spool does not move, and when $ro_i < 0$ that means the spool in the water distributor moves ro_i seconds in the direction of decreasing the overflow area.

E. COMBINING REINFORCEMENT LEARNING ALGORITHM DESIGN AND IMPLEMENTATION

SAC algorithm is a model-free deep reinforcement learning algorithm with maximum entropy, which can be trained in an offline environment, and is able to solve the reinforcement learning problem in discrete action space and continuous action space well [39], [40].

The reinforcement learning principle of ASWI algorithm (Automatic Stratified Water Injection) combined with PFC algorithm is shown in Figure 6, which is added to the original SAC algorithm as a pre-processing for water injection control. The policy network takes the state $\boldsymbol{s}$ of the environment as input, and each layer segment has four state attributes: nozzle opening x_i, depth of the injection device h_i, formation pressure p_i, and target injection volume ta_i, and the wellhead pressure p_0 is common to each layer segment. The number of input parameters is $1 + n \times 4$ for the water injection environment with n layer segments. For different number of layer segments, the hidden layer is n fully-connected layers, 64 neurons for one and two layer segments, and 128 neurons for the three-layer case, and LeakyReLU and Tanh are used as activation functions for the hidden layer and the output layer, respectively.

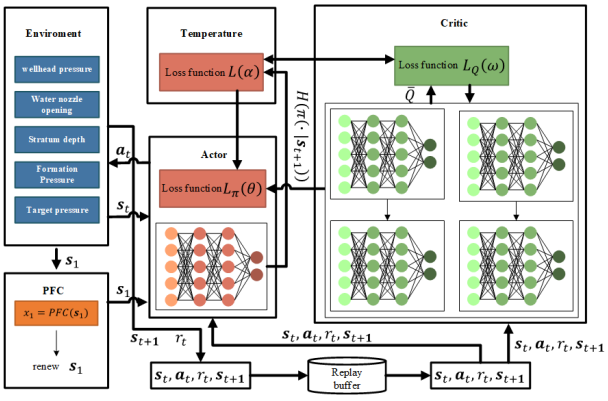


FIGURE 6. Schematic diagram of the layered water injection ASWI. Algorithm based on reinforcement learning.

Entropy denotes the degree of randomness with respect to a random variable, and entropy is defined as:

$$H(\pi(\cdot \mid \boldsymbol{s}_{t+1})) = -\log\pi(\boldsymbol{a}_{t+1} \mid \boldsymbol{s}_{t+1}) \tag{36}$$

The SAC algorithm maximizes the cumulative expected reward while making the strategy more stochastic, and the optimization objective of the strategy is defined as:

$$\pi^* = \arg\min_{\pi} \mathrm{E}_{\pi}\left[\sum_t r(\boldsymbol{s}_t, \boldsymbol{a}_t) + \alpha\mathcal{H}(\pi(\cdot \mid \boldsymbol{s}_t))\right] \tag{37}$$

where, $r \in (0, 1)$ the discount factor, which responds to the effect of future rewards on the current harvest; $\alpha \in (0, 1)$ the temperature coefficient, which controls the importance of entropy; and $\mathcal{H}(\pi(\cdot \mid \boldsymbol{s}_t))$ denotes the degree of randomness of the strategy π in state $\boldsymbol{s}$.

SAC uses two action-value functions $Q_{\omega_i}(\boldsymbol{s}_t, \boldsymbol{a}_t)$, and each time a Q network is used, the network with the smaller Q value is selected, thus mitigating the problem of having too high a Q. The loss function for any one of the functions Q is:

$$\begin{aligned} &L_Q(\omega) \\ &= \mathrm{E}_{(\boldsymbol{s}_t, \boldsymbol{a}_t, r_t, \boldsymbol{s}_{t+1}) \sim \mathcal{D}, \boldsymbol{a}_{t+1} \sim \pi_\theta(\cdot \mid \boldsymbol{s}_{t+1})}[\frac{1}{2}(Q_\omega(\boldsymbol{s}_t, \boldsymbol{a}_t) \\ &\quad - (r_t + \gamma(\min_{j=1,2} Q_{\omega_j^-}(\boldsymbol{s}_{t+1}, \boldsymbol{a}_{t+1})^2 - \alpha log\pi(\boldsymbol{a}_{t+1} \mid \boldsymbol{s}_{t+1})))] \end{aligned} \tag{38}$$

where, $\mathcal{D}$ is the data collected by the strategy in the past, $Q_{\omega_j^-}(\boldsymbol{s}_{t+1}, \boldsymbol{a}_{t+1})$ is approximated using the target network of Q. $Q_{\omega_j^-}$ is the target Q network with parameter ω^-, and the updating method is denoted as:

$$\omega_j^- \leftarrow \tau\omega_j + (1 - \tau)\omega_j^- \tag{39}$$

where, τ is the learning rate, the target network is updated by assigning a weighted average of the original target network and the corresponding Q-network after iterative learning.

The Q-network is updated using gradient descent and the gradient expression for the Q-network is:

$$\nabla_\omega L_Q(\omega) = \sum \frac{\nabla_\omega Q_\omega}{|\mathcal{B}|}(Q_\omega(\boldsymbol{s}_t, \boldsymbol{a}_t) - y_i) \tag{40}$$

where, $\mathcal{B}$ denotes a fixed batch size of samples $\mathcal{B}$ selected from the buffer $\mathcal{D}$. For each sample the y_i expression is computed using the target network as:

$$y_i = r_i + \gamma \min_{j=1,2} Q_{\omega_j^-}(\boldsymbol{s}_{i+1}, \boldsymbol{a}_{i+1}) - \alpha\log\pi_\theta(\boldsymbol{a}_{i+1} \mid \boldsymbol{s}_{i+1}) \tag{41}$$

The policy network is a state-to-action mapping, and the policy π is updated by minimizing the scatter, and the loss function of the policy π is denoted as:

$$L_{\pi(\theta)} = \mathrm{E}_{\boldsymbol{s}_t \sim \mathcal{D}, \boldsymbol{a}_t \sim \pi_\theta}\left[\alpha\log(\pi_\theta(\boldsymbol{a}_t, \boldsymbol{s}_t)) - Q_\omega(\boldsymbol{s}_t, \boldsymbol{a}_t)\right] \tag{42}$$

Since the process of sampling a Gaussian distribution is not derivable, the SAC algorithm uses a reparameterization to make the sampling process derivable for the policy function, which is denoted by the policy function $\boldsymbol{a}_t$:

$$\boldsymbol{a}_t = f_\theta(\epsilon_t; \boldsymbol{s}_t) \tag{43}$$

where, ϵ_t is a noisy random variable, and considering both Q-functions, the loss function of the rewrite strategy is:

$$L_\pi(\theta) = \mathrm{E}_{s_t\sim\mathcal{D},\epsilon_t\sim\mathcal{N}}[\alpha\log(\pi_\theta(f_\theta(\epsilon_t;s_t)\,|\,s_t)) - \min_{j=1,2} Q_{\omega_j}(s_t, f_\theta(\epsilon_t;s_t))] \tag{44}$$

The expression for the gradient $\nabla_\theta L_\pi$ of the policy network at time slot t is

$$\nabla_\theta L_\pi(\theta) = \sum \frac{1}{|\mathcal{B}|}\nabla_\theta \alpha\log(\pi_\theta(\boldsymbol{a}_t,\boldsymbol{s}_t)) + \left(\nabla_{\boldsymbol{a}_t}\alpha\log(\pi_\theta(\boldsymbol{a}_t,\boldsymbol{s}_t)) - \nabla_{\boldsymbol{a}_t}Q(\boldsymbol{s}_t,\boldsymbol{a}_t)\right)\nabla_\theta f_\theta(\epsilon_t;\boldsymbol{s}_t) \tag{45}$$

The temperature coefficient of entropy is very important in the SAC algorithm, and different sizes of temperature coefficients are chosen in different states. In order to automatically adjust the temperature coefficient of entropy, the SAC algorithm constructs an optimization problem with constraints defined as:

$$\max_\pi \mathrm{E}_\pi\left[\sum_t r(\boldsymbol{s}_t,\boldsymbol{a}_t)\right] \text{s.t.}\mathrm{E}_{(\boldsymbol{s}_t,\boldsymbol{a}_t)\sim\rho_\pi}\left[-\log(\pi_t(\boldsymbol{a}_t\,|\,\boldsymbol{s}_t))\right] \geq \mathcal{H}_0 \tag{46}$$

The loss function for α at time t can be obtained by transforming Equation (45) into a dyadic problem via the Lagrangian dyadic method:

$$L(\alpha) = \mathrm{E}_{\boldsymbol{s}_t\sim\mathcal{D},\boldsymbol{a}_t\sim\pi(\cdot\,|\,\boldsymbol{s}_t)}\left[-\alpha\log\pi(\boldsymbol{a}_t\,|\,\boldsymbol{s}_t) - \alpha\mathcal{H}_0\right] \tag{47}$$

where, $\mathcal{H}_0$ denotes the minimum policy entropy threshold.

IV. SIMULATION RESULTS AND ANALYSIS

In this chapter, simulation experiments are conducted to verify that ASWI is better than SAC algorithm and PFC algorithm in terms of convergence speed, training time, success rate, and water injection error under the same parameters and environment.

A. SIMULATION EXPERIMENTAL DESIGN

The PyTorch and the Python language were used to compile the PFC, SAC, and ASWI algorithms based on the models in Sections II-C and III-E, while the network models in Section IV-B were trained on a single 6GB RAM GPU (1660plus) and a 12400f CPU. The SAC and ASWI algorithms used the same training parameters, with the number of training steps set to 500, and the stopping condition of a single iteration exceeding 200 steps or a water injection error is less than 2%.

Since the algorithm is based on the SAC framework with both actor and critic networks, the network parameters are shown in Table 1:

Hyperparameters are the parameters that affect the performance, learning speed, convergence quality, robustness, and generalization ability of machine learning models, and need to be set before training. Through a large number of experiments, we adjust the hyperparameters to make the reinforcement learning model achieve better performance. The hyperparameter settings are shown in Table 2.

Algorithm 3 Automatic Stratified Water Injection

Initialization:
Randomize the network parameters ω_1, ω_2, and θ, initialize the Critic networks $Q_{\omega 1}(\boldsymbol{s},\boldsymbol{a})$, $Q_{\omega 2}(\boldsymbol{s},\boldsymbol{a})$, and the Actor network $\pi_\theta(\boldsymbol{s})$, copy the same parameters $\omega_1^- \leftarrow \omega_1$, $\omega_2^- \leftarrow \omega_2$, and $\theta^- \leftarrow \theta$, and initialize the target networks $Q_{\omega_1^-}$, $Q_{\omega_2^-}$, and π_{θ^-}.

Initialize the experience pool D.
for $e = 1 \rightarrow T$ **do**
Get environment initialization state $\boldsymbol{s}_t$.
Algorithm 1 $\boldsymbol{s}_t$ = PFC $(\boldsymbol{s}_t)$.
for $t = 1 \rightarrow T$ **do**
Choose the action $\boldsymbol{a}_t = \pi_\theta(\boldsymbol{s}_t)$ according to the current policy.
Execute $\boldsymbol{a}_t$, get reward r_t, and the state of the environment changes to $\boldsymbol{s_{t\mathcal{C}1}}$
$\mathcal{D} \leftarrow \mathcal{D} \cup \{(\boldsymbol{s}_t,\boldsymbol{a}_t,r(\boldsymbol{s}_t,\boldsymbol{a}_t),\boldsymbol{s}_{t+1})\}$
for $k = 1 \rightarrow K$ **do**
Sample B tuples from D
$\omega_j \leftarrow \omega_j - \lambda_Q\nabla_\omega L_Q(\omega_j)$ forj $\in \{1,2\}$
$\theta \leftarrow \theta - \lambda_\pi\nabla_\theta L_\pi(\theta)$
$\alpha \leftarrow \alpha - \lambda\nabla_\alpha L_\alpha(\alpha)$
$\omega_j^- \leftarrow \tau\omega_j + (1-\tau)\omega_j^-$ forj $\in \{1,2\}$
end for
end for
end for

TABLE 1. Hyperparameter setting of ASWI algorithm.

Parameter name	1layer	2layers	3layers
Number of input neurons to the actor network	6	11	16
Number of input neurons to the critic network	7	13	19
Actor network hidden layer neural network number	64	64	128
Critics network hidden layer neural network number	64	64	128
Number of neurons in the output layer of the actor network	1	2	3
Number of neurons in the output layer of the critic	1	2	3
Actor Network Learning Rate	$5e^{-4}$	$3e^{-3}$	$7e^{-3}$
Critic Network Learning Rate	$5e^{-4}$	$3e^{-3}$	$7e^{-3}$
activation function	LeakyReLU		

TABLE 2. Hyperparameter setting of ASWI algorithm.

Parameter name	1 layer	2 layers	3 layers
Alpha network learning rate	$5e^{-4}$	$3e^{-15}$	$3e^{-30}$
Training period	500	500	500
Discount factor	0.9	0.9	0.9
Soft update parameter	0.005	0.005	0.005
Experience replay buffer size	$5e^5$	$5e^5$	$5e^5$
Minimal sample	$2e^3$	$5e^3$	$6e^3$
Number of samples extracted	512	512	1024

The reinforcement learning model under the random water injection environment of 1-3 layer segments was simulated and trained several times through the simulation platform respectively, and some parameters of the water injection environment are shown in Table 3.

B. COMPARATIVE ANALYSIS OF REWARDS FOR TRAINING RESULTS OF WATER INJECTION ALGORITHMS

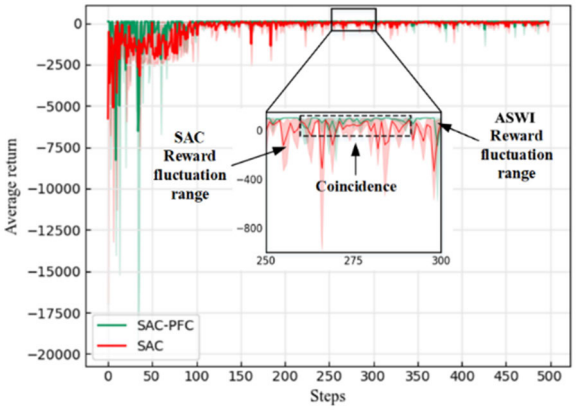
The reward curves can reflect the learning of ASC algorithm and ASWI algorithm during the training process, so the reward curves of the two algorithms under 500 steps of training were explored, with the horizontal coordinate being the number of training steps and the vertical coordinate being the reward value, as shown in Figure 7.

(1) As shown in Figure 7(a), the fluctuation range of the reward value of ASWI and SAC algorithms is closer to the range near 0 and the convergence speed is almost the same in the One-layer segment environment, and the fluctuation of SAC algorithm is slightly larger than that of ASWI algorithm, and the maximum reward value is basically the same as that of ASWI algorithm, as can be seen from the local zoomed-in graph.

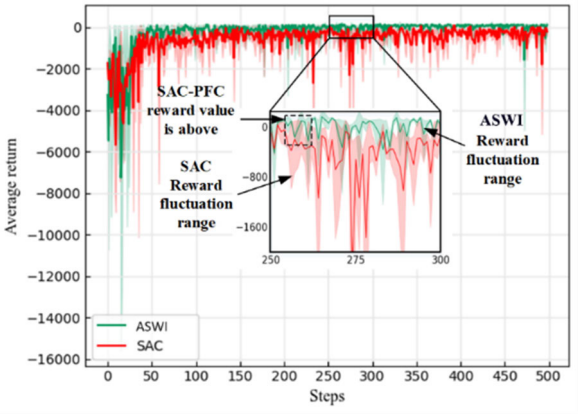
(2) As shown in Figure 7(b), the reward value of SAC algorithm is smaller than that of ASWI algorithm under the water injection environment of the Two-layer segment environment, and from the local zoomed-in figure, it can be seen that the reward value fluctuation range of SAC algorithm is larger than that of ASWI algorithm, with most values fluctuating around −2000, and some of the reward values of ASWI algorithm fluctuates more, and the maximum fluctuation value is around −1000, comparing with the SAC algorithm, ASWI has a Compared with SAC algorithm, ASWI has better reward value stability.

(3) As shown in Figure 7(c), the ASWI reward value is larger than the SAC algorithm in the Three-layer segment environment, the SAC algorithm reward value fluctuates in a larger range, with most of the data fluctuating around −2000, and the ASWI algorithm rewards the largest fluctuating value around 0, which has a better reward stability.

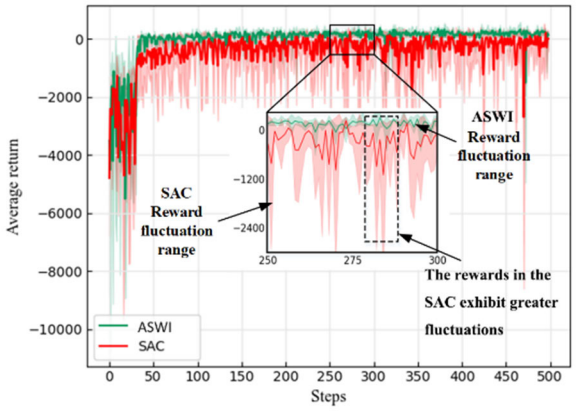
The two algorithms are significantly different in reward value and fluctuation range as shown in Figure 7. The SAC



(a)One layer reward curve



(b)Two layers reward curves



(c)Three layers reward curves

FIGURE 7. SAC algorithm and ASWI algorithm reward curves.

algorithm reward value fluctuation range is larger indicating that the single-step adjustment process does not reach the target fast enough and therefore obtains more penalties for the number of adjustment steps and the adjustment error. From the SAC algorithm reward curve coverage, it can be seen that the frequency and range of reward data fluctuation increases significantly with the number of layer segments. In the One-ayer segment environment, the reward value has some data fluctuating around −400 near 0, some data around −2000 in the Two-layer segment, and a large amount of data

TABLE 3. Partial test environment data.

1-th layer				2-th layer				3-th layer			
volume (m^3/d)	openin g (%)	stratigraphic pressure (MPa)	depth (m)	volume (m^3/d)	opening (%)	stratigraphic pressure (MPa)	depth (m)	volume (m^3/d)	openin g (%)	stratigraphic pressure (MPa)	depth (m)
19.50	66.36	22.43	1052.27	-	-	-	-	-	-	-	-
113.1	85.42	24.36	1034.78	-	-	-	-	-	-	-	-
57.14	31.04	20.47	1020.1	-	-	-	-	-	-	-	-
119.88	64.92	22.49	1095.97	-	-	-	-	-	-	-	-
71	40.83	23.42	1018.42	-	-	-	-	-	-	-	-
48.84	37.69	22.88	1027.58	9.94	9.53	23.72	1115.64	-	-	-	-
46.63	42.59	22.7	1010.79	30.59	28.37	24.25	1172.19	-	-	-	-
49.93	43.13	23.1	1020.8	33.13	29.04	23.87	1100.27	-	-	-	-
116.63	73.5	19.07	1059.81	19.36	13.72	19.72	1127.22	-	-	-	-
83.25	47.15	19.15	1044.8	83.33	47.21	19.86	1118.56	-	-	-	-
87.43	76.47	24.87	1089.67	52.83	47	25.01	1105.09	43.06	38.6	26.56	1266
107.76	73.36	23.95	1048.27	0.05	0.25	25.28	1187.5	95.25	65.32	25.91	1253.07
32.76	31.14	24.05	1025.75	9.86	10.97	25.02	1126.52	49.06	46.01	25.94	1222.91
2.03	4.09	23.48	1028.85	72.33	76.81	25.04	1191.14	41.47	44.9	25.88	1278.83
97.29	65.55	20.04	1082.66	83.82	56.55	20.22	1101.25	77.96	52.68	21.61	1246.18

appearing around −2000 in the Three-layer segment. The larger reward fluctuation of SAC indicates that the reward acquisition is unstable. The reward value of ASWI algorithm tends to stabilise at around 0 in the Three-layer segment complexity environments after 100 steps of training. fluctuate around 0. From the coverage area of the reward curve of the ASWI algorithm in Fig. 7, it can be seen that the ASWI algorithm has a larger reward value and better stability of the reward value compared to the SAC algorithm.

C. COMPARATIVE ANALYSIS OF ALGORITHM TRAINING TIME

We comparatively studied the time required to complete 500 steps of training for the two algorithms in different environmental complexities, as shown in Table 4 and Figure 8. Where Figure 8 horizontal coordinates are 1-3 layer segment environments, the left y-axis corresponds to a bar graph indicating training time, and the right y-axis corresponds to a line graph indicating the rate of improvement in the training time of the ASWI algorithm.

TABLE 4. Training time for SAC and ASWI algorithms.

Algorithm name	1 layer	2 layers	3 layers	Average
SAC	5.20min	21.38min	48.8min	25.12min
ASWI	2.23min	12.15min	35.47min	16.62min
promotion rate	57.12%	41.39%	27.32%	41.94%

As can be seen from Table 4, under One-layer segment water injection environment, the training time of SAC and ASWI algorithms are 5.2min and 2.23min respectively, and the training time enhancement rate of ASWI algorithm is 57.12%; under Two-layer segment water injection environment, the training time of SAC and ASWI algorithms are 21.38min and 12.15min respectively, and the training time enhancement rate of ASWI algorithm time enhancement rate is 41.39%; in the Three-layer segment water injection environment, the training time of SAC and ASWI algorithms are 48.8min and 35.47min respectively, and the training time enhancement rate of ASWI algorithm is 27.32%.

As can be seen in Figure 8, the training time of SAC and ASWI algorithms gradually increases with the increase in the complexity of the environment, and the training time enhancement rate of the ASWI algorithm gradually shrinks into a linearly decreasing trend. The ASWI algorithm has an obvious advantage in training time compared to the SAC algorithm, and it performs well especially in the case of low environmental complexity. It can also be found that the training time of reinforcement learning for hierarchical water injection increases linearly with the increase of the complexity of the water injection environment.

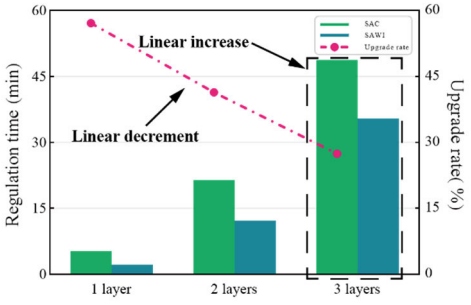


FIGURE 8. Comparison of training time for layered water injection regulation algorithms.

The ASWI algorithm has an obvious advantage in training time compared with the SAC algorithm, and especially performs well in the case of low environmental complexity. Meanwhile, the training time of reinforcement learning for layered water injection increases linearly with the increase of the complexity of the water injection environment.

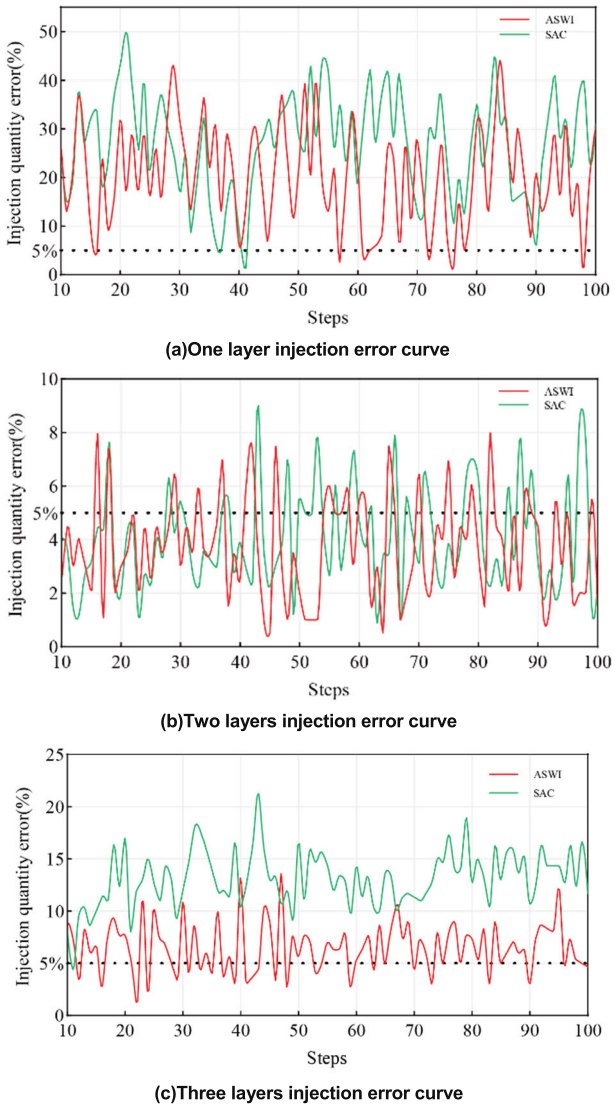


FIGURE 9. Variation curve of water injection error.

D. COMPARATIVE ANALYSIS OF INJECTION ERRORS IN LAYER SECTIONS

We compare and analyze the error change curves in a particular set of environments, as shown in Figure 9, where the horizontal coordinate is the number of computational steps and the vertical coordinate is the layer-segment regulated water injection error.

(1) As shown in Figure 9(a), the error range of the SAC algorithm is between 0% and 50% and the error range of the ASWI algorithm is between 0% and 42% for the One-layer segment environment, the fluctuation range of the ASWI is smaller and the ASWI algorithm is closer to the target injection amount when 5% is the injection target.

(2) As shown in Figure 9(b), the error fluctuation range of SAC algorithm is between 0% and 9% and the error fluctuation range of ASWI algorithm is between 0 and 8% in the Two-layer segment environment, and the two algorithms have similar performance in controlling the water injection error.

(3) As shown in Figure 9(c), the error range of the SAC algorithm is between 5% and 20% and the error range of the ASWI algorithm is between 2% and 10% for the Three-layer segment environment, and the error of the ASWI algorithm is significantly smaller than that of the SAC algorithm, and the error of the SAC algorithm gradually increases with the increase of the environmental complexity, and the ASWI algorithm shows a more excellent performance in controlling the water injection error.

As can be seen from Figure 9, under One-layer segment, SAC algorithm and ASWI algorithm have almost similar error control ability, under Two-layer segment ASWI algorithm has a smaller error range, and under Three-layer segment SAC error is much larger than ASWI algorithm. ASWI has a good performance in the water injection scenario with different complexity, and all of them can keep the water injection error around 5%, and has a more excellent performance in the water injection error control. It has more excellent performance in water injection error control.

The distribution of training errors of analyzing PFC, SAC, and ASWI algorithms under different environmental complexity is shown in Figure 10, where the horizontal coordinate is the layer segment regulation error and the vertical coordinate indicates the control algorithms under different environmental complexity.

(1) As shown in Figure 10(a), the quartiles of PFC, SAC, and ASWI for the One-layer segment environment are about 9%, 4%, and 4%, the quartiles are about 1%, 0%, and 0%, and the medians are about 4%, 2%, and 2%. From the One-layer segment error box plots, it can be seen that the quartile error values of the PFC algorithm are larger and have multiple deviations compared to SAC and ASWI.

(2) As shown in Figure 10(b), the quartiles of PFC, SAC, and ASWI for the Two-layer segment environment are about 8%, 7%, and 5%, the quartiles are about 1%, 0%, and 0%, and the medians are about 4%, 2%, and 2%. From the Two-layer segment error box plots, it can be seen that the PFC algorithm

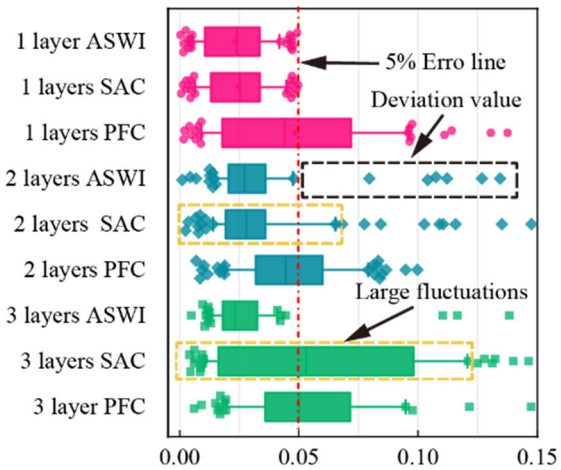


FIGURE 10. Comparison of algorithmic water injection error distributions.

TABLE 5. Data on average error and success rate for injection quantities.

Algorithm Name	average error of injection quantities (%)				Water injection success rate (%)			
	1 layer	2 layers	3 layers	average	1 layer	2 layers	3 layers	Average
PFC	4.85%	4.53%	11.74%	7.04%	56%	62%	49%	55.67%
SAC	2.47%	3.40%	6.25%	4.04%	100%	86%	50%	78.33%
ASWI	2.33%	3.56%	3.22%	3.03%	100%	91%	94%	95%

has a higher quartile and one-quartile error than SAC and ASWI, the SAC algorithm has a higher quartile error than the ASWI algorithm, SAC algorithm has more number of deviation points than ASWI algorithm and the range of deviation is more.

(3) As shown in Figure 10(c), the quartiles of PFC, SAC, and ASWI are about 9%, 12.5%, and 4%, the quartiles are about 2%, 1%, and 1%, and the medians are about 5%, 5%, and 2.5% for the Three-layer segment environment. From the Three-layer segment error box plots it can be seen that the error range of SAC algorithm is greater than ASWI and PFC algorithms and the quartiles of SAC algorithm is greater than ASWI and PFC algorithms and ASWI algorithm is less than SAC and PFC algorithms.

The SAC algorithm has a small range of error distribution in the one and two layer segments with median around 2.5%, in the Three-layer segment, the range of error distribution and median significantly increases median around 5%.The ASWI algorithm has quartiles below 5% in complex environments, with no off-points in the One-layer segment, and partial off-points in the two and three layer segments, under complex environments, the ASWI is still able to maintain better error control.

The average error and success rate of injection volume are counted to compare the error performance of different algorithms as shown in Table 5, and the target is considered to be reached when the injection error is less than 5%. Combined with Table 5, the average error and dispensing success rate histograms are plotted as shown in Figure 11 to show the injection error performance of different algorithms under different environmental complexity, the horizontal coordinate is the number of water injection environmental layer segments, and the histogram and line graph represent the average injection error and injection success rate of different algorithms, respectively.

(1) The average injection error of the three algorithms in the One-layer segment environment is less than 5%, and the average values of the PFC, SAC, and ASWI algorithms are 4.85%, 2.47%, and 2.33%, respectively, and the success rates are 56%, 100%, and 100%, respectively, and the average injection errors and success rates of the SAC and ASWI algorithms are almost the same and lower than those of the PFC algorithm.

(2) The average injection error of the three algorithms in the Two-layer segment environment is less than 5%, and the average values of the PFC, SAC, and ASWI algorithms are 4.53%, 3.40%, and 3.56%, respectively, with the success rates of 62%, 86%, and 91%, respectively, and the average

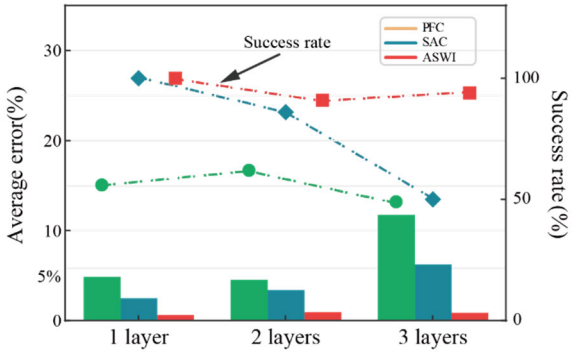


FIGURE 11. Bar chart of average error and success rate in annotation.

injection values of the SAC and ASWI algorithms are close to each other and smaller than the average injection error of the PFC algorithm, and the success rate of the ASWI algorithm is higher than that of the PFC and SAC algorithm.

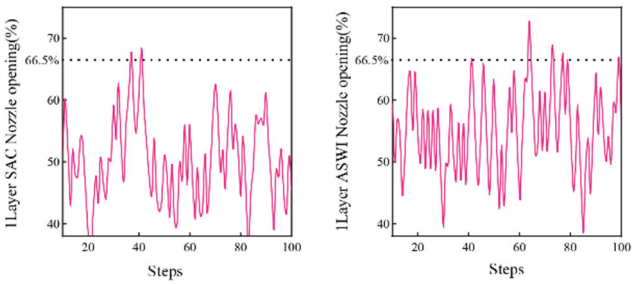
(3) The average injection error of ASWI algorithm is less than 5% in the Three-layer segment environment, the average injection error of ASWI algorithm is 3.03% less than PFC and SAC algorithms, SAC and PFC algorithms have similar success rate close to 50%, and ASWI algorithm has 94% success rate more than PFC and SAC algorithms.

In terms of the average error and success rate under different environmental complexity, the average water injection error of PFC and SAC algorithms increases with the increase of environmental complexity, and the success rate of SAC algorithm performs well at 2.47% in the One-layer segment environment, and PFC algorithm can keep the error within 5% in the One-layer segment and Two-layer segment environment. However, with the increase of environment complexity, the success rate of SAC is obviously decreasing, and the PFC algorithm is around 50%.The average error and success rate of ASWI in different environments are 3.03% and 95% respectively, which are better than PFC and SAC algorithms, and fluctuation is small with the increase of environment complexity.

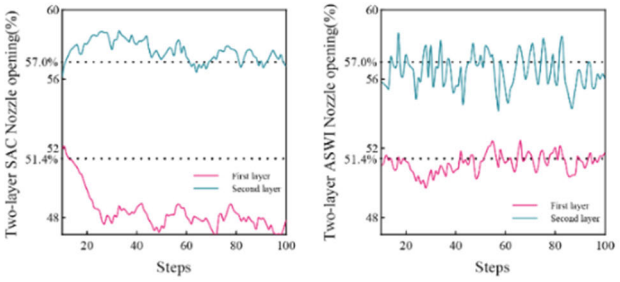
E. INJECTIONS MEAN ERROR AND SUCCESS RATE DATA SHEET

The adjustment steps of different algorithms were analyzed, as shown in Figure 12, for the change of water nozzles of the water injection device under the 1-3 layer section, with the horizontal coordinate as the number of adjustment steps and the vertical coordinate as the percentage of nozzle opening.

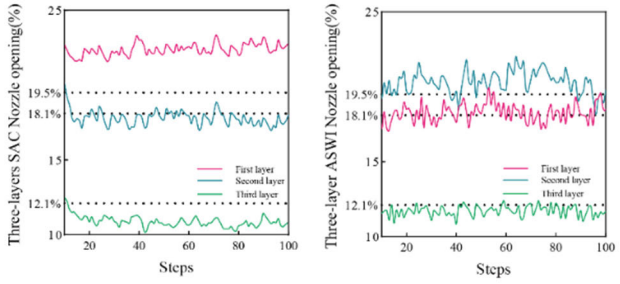
(1) As shown in Figure 12(a), the nozzle opening for an injection error of 0% is approximately 66.5% in the One-layer



（a）Nozzle opening variation curve for one layer



（b）Nozzle opening variation curve for two layers



（c）Nozzle opening variation curve for three layers

FIGURE 12. Nozzle opening variation curve for water injection device.

segment environment. In the ASWI algorithm, the spout openings occur more densely near 66.5%, the SAC algorithm has some data near 66.5% spout openings, and the ASWI algorithm is more densely near the optimal openings.

(2) As shown in Figure 12(b), the nozzle openings with 0% injection error for the first and second layer segments are approximately 51.4% and 57.0% in the Two-layer segment environment. In the SAC algorithm, the first layer segment gradually converges to 57.0% of the water nozzle opening after 60 steps, and the second layer segment gradually converges to 46% of the water nozzle opening, which is a large gap from the optimal opening. In the ASWI algorithm, most of the nozzle opening data in the first segment is below 57.0% and has a large gap, and the nozzle opening in the second segment fluctuates around the optimal target opening.

(3) As shown in Figure 12(c), the nozzle openings with 0% injection error for the first and second layer segments are approximately 18.1%, 19.5%, and 12.1% in the Three-layer segment environment. In the SAC algorithm, the first nozzle opening is approximately 22% above the optimal opening approximately, and the second and third nozzle openings are approximately 17% and 10% fluctuating below the optimal opening. In the ASWI algorithm, the first and second layers have denser fluctuations near the optimal spout opening, and the third layer spout opening is closer to the optimal value and fluctuates below.

The SAC algorithm was able to have some of the data appear near the optimal opening in the One-layer segment environment, with large offsets in both two and three layer segments. The ASWI algorithm was able to remain near the spigot opening as the complexity of the environment increased, with denser data in the vicinity of the optimal opening, representing a higher probability of a higher probability of the ASWI algorithm with fewer tuning steps in the same environment.

Statistics on the distribution of the number of adjustment steps for each algorithm under different environmental complexity are shown in Figure 13, where the horizontal coordinate is the interval from 1-100 steps and the vertical coordinate is the algorithm under different environmental complexity.

(1) In the One-layer segment environment, the SAC algorithm and the ASWI algorithm reached the goal in the 1-10 step interval at 60% and 91%, respectively. The percentage of reaching the target within the 1-5 step interval was 32% and 82%, respectively. The mean values of the adjusted steps were 9.34 and 3.43 steps, respectively, and the ASWI algorithm outperformed the SAC algorithm in reaching the goal within the 1-10 step interval.

(2) In the Two-layer segment environment, the SAC algorithm and the ASWI algorithm reach the goal within 1-10 steps in 72% and 89%, respectively. The percentage of reaching the target within 95-100 steps is 24% and 6%, respectively. The average values of adjustment steps are 25.91 and 9.52 steps, respectively, and the ASWI algorithm has fewer adjustment steps in both layer segment environments compared to the SAC algorithm.

(3) In the Three-layer segment environment, the SAC algorithm and the ASWI algorithm reach the goal within 1-5 steps in 55% and 85%, respectively. The percentage of reaching the target within 1-10 steps is 33% and 76%, respectively. The percentage of reaching the target within 95-100 steps is 54% and 9%, respectively. The average values of adjustment steps are 58.97 and 16.58 steps, respectively, so the adjustment steps of ASWI are smaller than those of SAC algorithm in three individual layer segment environments.

The average percentage of SAC algorithm and ASWI algorithm reaching the goal within 1-5 steps is 43.00% and 73.33% respectively in 1-3 layer segment environment. The average percentage of reaching the goal in the range of 1-10 steps is 55.00% and 85.33%, respectively. As shown

TABLE 6. Average step data.

Algorithm Name	1 layer	2 layers	3 layers	Average
SAC	9.34	25.91	58.97	31.41
ASWI	3.43	9.52	16.58	9.84

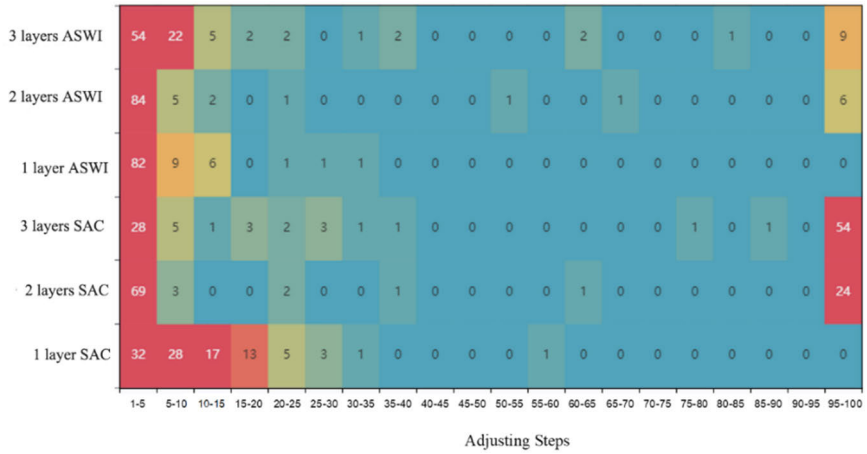


FIGURE 13. Heatmap of adjustment step distribution.

in Table 6, the average number of adjustment steps for SAC algorithm and ASWI algorithm for different environment complexity are 34.41 and 9.84 steps respectively. The ASWI algorithm outperforms the SAC algorithm in the number of adjustment steps to reach the target injection faster.

V. CONCLUSION

In order to solve the problem of difficult to control the flow rate of water nozzles due to the complexity and incompleteness of stratified water injection layer segments in the water injection process, this paper proposes a reinforcement learning-based ASWI automatic deployment algorithm. The water injection device is able to perform actions continuously in a nonlinear environment, and regulate the water nozzle opening of each layer segment according to the environmental information in order to achieve the injection target of different layer segments. In this paper, a gym-based simulation environment is constructed to simulate the water injection process in multi-layer segments using the local loss coefficient as a nonlinear parameter, and the environment is used to train and validate the control performance of the algorithm in a nonlinear environment.

Based on the hydrodynamics of layered water injection column, establish the water nozzle control model of water injection device under constant pressure state. Build a PyTorch-based algorithmic program based on the nonlinear water injection environment, train the intelligent body, and design a reasonable reward function with the fastest tuning speed and the smallest injection error as the goal. Comparison of the performance of PFC algorithm, SAC algorithm and ASWI algorithm is verified by arithmetic simulation, and the results show that the ASWI algorithm shortens the training time by an average of 41.94%, the regulation error of each layer segment is under 5% in 1-3 layer segment, and the average success rate of the regulation of the layer segments to reach the injection target is 95%, and there is a probability of 85% to reach the layer segment injection target in 1-10 steps, and the ASWI algorithm has better performance in terms of training reward, training time, layer water injection error and the number of tuning steps outperforms the PFC algorithm and SAC algorithm. This method makes up for the lack of control of layered water injection devices in the field of water injection and has important application value for oil development.

Future research work will be devoted to the analysis of nonlinear influences on the injection environment, modeling of the constant flow injection environment, and deep learning-based prediction of the injection environment These topics will be considered as future extensions of this work.

REFERENCES

[1] Z. Zhang, H. Li, and D. Zhang, "Water flooding performance prediction by multi-layer capacitance-resistive models combined with the ensemble Kalman filter," *J. Petroleum Sci. Eng.*, vol. 127, pp. 1–19, Mar. 2015.

[2] H. Liu, G. Xiao, F. Sun, X. Pei, H. Hu, H. Geng, and L. Li, "A new concentric zonal water injection technique for highly-deviated wells," *Petroleum Explor. Develop.*, vol. 42, no. 4, pp. 560–566, Aug. 2015.

[3] J. Zhang, B. Deng, C. Hu, J. Chang, X. Li, H. Li, and D. He, "Computation method for water influx in different layers of natural edge water," *Petroleum Explor. Develop.*, vol. 43, no. 5, pp. 825–831, Oct. 2016.

[4] Y. Qu, "Simulation and optimization of oilfield water injection system," M.S. thesis, Dept. Comput. Math., Jilin Univ., Changchun, China, 2003.

[5] H. Zhang, "The research of simulation and operating optimization on oilfield water injection system," M.S. thesis, Dept. Chem. Eng., Wuhan Univ. Technol., Wuhan, China, 2010.

[6] L. Li, "The research of the interference regulation between layers in oilfield injection wells and application," M.S. thesis, Dept. Earth Sci., Zhejiang Univ., Hangzhou, China, 2011.

[7] H. Liu, L. Zheng, J. Yu, E. Ming, Q. Yang, D. Jia, and G. Cao, "Development and prospect of downhole monitoring and data transmission technology for separated zone water injection," *Petroleum Explor. Develop.*, vol. 50, no. 1, pp. 191–201, Feb. 2023.

[8] H. Liu, X. Pei, D. Jia, F. Sun, and T. Guo, "Connotation, application and prospect of the fourth-generation separated layer water injection technology," *Petroleum Explor. Develop.*, vol. 44, no. 4, pp. 644–651, Aug. 2017.

[9] C. Chen, Y. Ming, H. Xiaodong, and Z. Jian, "Water flooding performance prediction in layered reservoir using big data and artificial intelligence algorithms," in *Proc. Day 3 Wed*. Abu Dhabi, UAE: SPE, Nov. 2019, Art. no. D032S184R002.

[10] L. Zhou, G. Han, F. Wang, R. Liu, X. Meng, and C. Niu, "Determining method of nozzle size used for concentric layered water injection," *Oil Drilling Prod. Technol.*, vol. 37, no. 5, pp. 95–99, 2015.

[11] G. Jiang, H. Li, X. Wang, and J. Mu, "The optimum selection method for the layered injection allocation nozzle in heterogeneous reservoirs," *China Petroleum Mach.*, vol. 40, no. 1, pp. 9–12, 2012.

[12] J. Wang, X. Li, W. Jiao, X. Meng, and P. Lu, "Method used to select water injection choke for 2-stage horizontal well," *Drilling Prod. Technol.*, vol. 40, no. 5, pp. 43–45, 2017.

[13] S. H. Strogatz, *Nonlinear Dynamics and Chaos*. Boulder, CO, USA: Westview Press, 1994.

[14] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, Nov. 2017.

[15] J. Zhang, Y. Dong, J. Zhu, J. Zhu, M. Kuang, and X. Yuan, "Improving transferability of 3D adversarial attacks with scale and shear transformations," *Inf. Sci.*, vol. 662, Mar. 2024, Art. no. 120245.

[16] Z. Zhang, D. Zhang, and R. C. Qiu, "Deep reinforcement learning for power system applications: An overview," *CSEE J. Power Energy Syst.*, vol. 6, no. 1, pp. 213–225, Mar. 2020.

[17] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," *IEEE Trans. Neural Netw.*, vol. 9, no. 5, p. 1054, Sep. 1998.

[18] V. Tsounis, M. Alge, J. Lee, F. Farshidian, and M. Hutter, "DeepGait: Planning and control of quadrupedal gaits using deep reinforcement learning," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 3699–3706, Apr. 2020.

[19] N. Alhousani, M. Saveriano, I. Sevinc, T. Abdulkuddus, H. Kose, and F. J. Abu-Dakka, "Geometric reinforcement learning for robotic manipulation," *IEEE Access*, vol. 11, pp. 111492–111505, 2023.

[20] Z. Iklassov, I. Sobirov, R. Solozabal, and M. Takáč, "Reinforcement learning approach to stochastic vehicle routing problem with correlated demands," *IEEE Access*, vol. 11, pp. 87958–87969, 2023.

[21] H. Qi, Z. Hu, H. Huang, X. Wen, and Z. Lu, "Energy efficient 3-D UAV control for persistent communication service and fairness: A deep reinforcement learning approach," *IEEE Access*, vol. 8, pp. 53172–53184, 2020.

[22] H. Liu and W. Wu, "Federated reinforcement learning for decentralized voltage control in distribution networks," *IEEE Trans. Smart Grid*, vol. 13, no. 5, pp. 3840–3843, Sep. 2022.

[23] C. Si, Y. Tao, J. Qiu, S. Lai, and J. Zhao, "Deep reinforcement learning based home energy management system with devices operational dependencies," *Int. J. Mach. Learn. Cybern.*, vol. 12, no. 6, pp. 1687–1703, Jun. 2021.

[24] O. Vinyals et al., "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, Nov. 2019.

[25] A. Perrusquía, W. Yu, and X. Li, "Multi-agent reinforcement learning for redundant robot control in task-space," *Int. J. Mach. Learn. Cybern.*, vol. 12, no. 1, pp. 231–241, Jan. 2021.

[26] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, Oct. 2017.

[27] G. Zhao, G. Ji, S. Yang, G. Lv, X. Du, and H. Guo, "Allocation mothed and calculation of layered injection rate of liquid control intelligent layered water injection process," *Fault Block Oil Gas Field*, vol. 28, no. 2, pp. 258–261, 2021.

[28] F. White, *Fluid Mechanics*. New York, NY, USA: McGraw-Hill, 2010.

[29] T. Fang, F. Ren, H. Liu, Y. Zhang, and J. Cheng, "Progress and development of particle jet drilling speed-increasing technology and rock-breaking mechanism for deep well," *J. Petroleum Explor. Prod. Technol.*, vol. 12, no. 6, pp. 1697–1708, Jun. 2022.

[30] S. Ishii, H. Fujita, M. Mitsutake, T. Yamazaki, J. Matsuda, and Y. Matsuno, "A reinforcement learning scheme for a partially-observable multi-agent game," *Mach. Learn.*, vol. 59, nos. 1–2, pp. 31–54, May 2005.

[31] K. Li, T. Zhang, R. Wang, W. Tan, H. He, and H. Huang, "Research reviews of combinatorial optimization methods based on deep resin for cement learning," *Acta Automa Matica Sinica*, vol. 47, no. 11, pp. 2521–2537, 2021.

[32] W. Wang, H. Chen, G. Li, L. Ling, and Y. Zhao, "Deep reinforcement learning for multi-depot vehicle routing problem," *Control Decis.*, vol. 37, no. 8, pp. 2101–2109, 2022.

[33] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1238–1274, Sep. 2013.

[34] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, May 1992.

[35] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," Jul. 2019, *arXiv:1509.02971*.

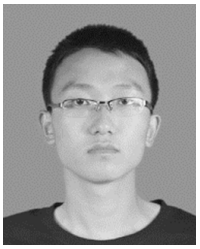
[36] B. D. Nichols, "Continuous action-space reinforcement learning methods applied to the minimum-time swing-up of the acrobot," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Hong Kong, Oct. 2015, pp. 2084–2089.

[37] T. Lesort, N. Díaz-Rodríguez, J.-F. Goudou, and D. Filliat, "State representation learning for control: An overview," *Neural Netw.*, vol. 108, pp. 379–392, Dec. 2018.

[38] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, Jul. 2018, pp. 1861–1870.

[39] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, "Soft actor-critic algorithms and applications," 2018, *arXiv:1812.05905*.

[40] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. Int. Conf. Mach. Learn.*, Jan. 2014, pp. 387–395.



JINZHAO HU was born in Hulunbuir, Inner Mongolia, China, in 1996. He received the bachelor's degree in mechanical engineering from Jilin Institute of Chemical Technology, China, in 2019. He is currently pursuing the Ph.D. degree in mechanical engineering with Northeast Petroleum University, Daqing, China.

His current research interests include intelligent design of water injection well systems and smart control methods for water injection devices.

FUSHEN REN was born in Liaoyang, Liaoning, China, in 1976. He received the B.S. degree in mechanical design, manufacture and automation from Northeast Petroleum University, in 1999, and the Ph.D. degree in mechanical and electronic engineering from Beijing University of Technology, Beijing, China, in 2009.

From 2009 to 2014, he was an Associate Professor with Northeast Petroleum University. Since 2014, he has been a Professor and the Doctoral Advisor with the School of Mechanical Science and Engineering, Northeast Petroleum University. He is the author of three books, more than 80 articles, and more than 30 inventions. His research interests include industrial robots and oil drilling equipment.

ZHI WANG was born in Jiuquan, Gansu, China, in 2000. He received the B.S. degree in automation from Harbin University of Science and Technology, Harbin, in 2022. He is currently pursuing the M.S. degree in control engineering with the Research Institute of Petroleum Exploration and Development, Beijing.

His current research interests include the design of water injection well communication systems and petroleum mechanics.

DELI JIA was born in Heilongjiang, China, in 1980. He received the bachelor's degree in mechanical design, manufacturing, and automation and the master's degree in applied chemistry from Harbin Institute of Technology, in 2002 and 2004, respectively, and the Ph.D. degree in mechanical manufacturing and automation from Harbin University of Science and Technology, in 2008.

From 2009 to 2012, he was an Associate Professor with Harbin University of Science and Technology, and then became a Professor from 2012 to 2013. Since 2013, he has been serving as a corporate expert and a professor-level Senior Engineer at the Research Institute of Petroleum Exploration and Development, Beijing. His research interests include oilfield water injection equipment and intelligent oilfield production.