Research

Study of Q-learning and deep Q-network learning control for a rotary inverted pendulum system

Zied Ben Hazem[1]

Received: 30 November 2023 / Accepted: 18 January 2024
Published online: 02 February 2024

Abstract
The rotary inverted pendulum system (RIPS) is an underactuated mechanical system with highly nonlinear dynamics and it is difficult to control a RIPS using the classic control models. In the last few years, reinforcement learning (RL) has become a popular nonlinear control method. RL has a powerful potential to control systems with high non-linearity and complex dynamics, such as RIPS. Nevertheless, RL control for RIPS has not been well studied and there is limited research on the development and evaluation of this control method. In this paper, RL control algorithms are developed for the swing-up and stabilization control of a single-link rotary inverted pendulum (SLRIP) and compared with classic control methods such as PID and LQR. A physical model of the SLRIP system is created using the MATLAB/Simscape Toolbox, the model is used as a dynamic simulation in MATLAB/Simulink to train the RL agents. An agent trainer system with Q-learning (QL) and deep Q-network learning (DQNL) is proposed for the data training. Furthermore, agent actions are actuating the horizontal arm of the system and states are the angles and velocities of the pendulum and the horizontal arm. The reward is computed according to the angles of the pendulum and horizontal arm. The reward is zero when the pendulum attends the upright position. The RL algorithms are used without a deep understanding of the classical controllers and are used to implement the agent. Finally, the outcome indicates the effectiveness of the QL and DQNL algorithms compared to the conventional PID and LQR controllers.

Article highlights

- Development of model-free reinforcement learning controllers for underactuated mechanical system.
- Utilization of QL and DQNL algorithms to enhance the control performance of a highly nonlinear system.
- This work represents an indispensable resource for researchers interested in advancing the field of robotic systems using the development of non-linear RL controllers.

Keywords Single-link rotary inverted pendulum (SLRIP) · PID · LQR · Q-learning (QL) · Deep-Q network learning (DQNL)

✉ Zied Ben Hazem, z.hazem@utb.edu.bh | [1]Department of Mechatronics Engineering, College of Engineering (COE), University of Technology Bahrain, Salmabad 18041, Bahrain.

1 Introduction

Many controllers have been developed to control mechatronics and robotic systems. The structures of controllers are mostly considered linear and nonlinear controllers. According to the existing research literature, linear controllers are summarized such as proportional (P), proportional-integral (PI), proportional derivative (PD), proportional integral derivative (PID), linear-quadratic regulator (LQR), linear-quadratic-gaussian (LQG), H-infinity (H∞) and H2-optimal control [1, 2]. Generally, linear controllers are based on the system linearized methods. Linearized models may cause instability and poor performance because they cannot represent the full dynamic of nonlinear systems. Linearization of the system is an approximation and cannot be considered as an appropriate foundation to develop and analyze robust control law designs. Additionally, linear controllers are extremely sensitive to external/internal disturbances that can affect the system output. The performance of linear controllers is completely related to the parameters of the controller to find optimal parameters for a complex nonlinear system [3]. The performance attained is related to the level of the system when it is linear around an operating point. Moreover, the application of linear controllers to any mechatronic system is related to its number of degrees of freedom and nonlinear dynamics complexity. The analysis of nonlinear systems is more complicated due to the high complexity of dynamics. Hence, the development of a sophisticated controller needs a deeper understanding of the system dynamics. During the last decade, different types of non-linear controllers have been developed such as sliding mode control [4], model-free controllers [5], adaptive nonlinear controllers, neural network (NN) control, hybrid PID with NN controllers, adaptive neuro-fuzzy and self-learning controllers [6]. Numerous nonlinear controllers are studied to involve all nonlinear system dynamics. Nonlinear controllers take into consideration all dynamic parameters of the system to achieve high control performance. For the implementation of a nonlinear controller, a linearization model and gains of states are not required. The control of RIPS is a challenge due to the complex nonlinear dynamical model of the system. RIPS always remains a control example of an underactuated mechanical system due to its importance in the field of control engineering. The RIPS has a high degree of nonlinearity and instability. RIPS is a very important system used for the development of balancing control systems such as legged robots, aerial, satellites, robots, two-wheeled transporters, drones and rockets [7].

In the last few years, deep learning (DL) has become a novel artificial intelligence methodology to make nonlinear controllers intelligent by adding self-learning algorithms in the robotic domain [8]. Artificial neural networks (ANNs) could enhance the self-learning of systems to get the required output. Many researchers related to the RIPS have found the effectiveness and robustness of controllers using the NN with fuzzy such as radial basis neuro-fuzzy [9] and fuzzy-based multi-layer feed-forward neural networks [10].

DL has made a vast contribution to solving the complexity of nonlinearity and instability of the systems. Moreover, DL has improved the control performance of systems by setting RL and control in one application. The DL algorithm was developed to control RIPS and industrial robot arms. QL is one of the best RL algorithms and it's usually used to control autonomous robots, car parking problems, quadruped robots, and walking robots [11, 12]. RL uses learning control policies to solve the problem by optimizing reward signals. Nevertheless, QL is used to estimate the value of actions by a maximization step that tends to overvalue rather than undervalue. In this overestimation problem, the agent chooses non-optimal operations for any state due to the large Q values of the agent. The RL algorithm updates the program according to the rewards and observations obtained from the environment to maximize the expected long-term reward [13]. RL is a modelization of how human beings learn by acting on the current state of the environment and obtaining rewards. Furthermore, deep reinforcement learning (DRL) is a combination of RL and DL. RL takes into consideration the errors of agent learning calculation to make decisions. DRL algorithms may take very large inputs and decide the best actions to perform the optimization of the objective. DRL combines DL into the solution which allows agents to make decisions from input data without taking into consideration the state space of the system. DRL has been used for a different application such as video games, language processing, computer vision, transportation, healthcare, finance and robotics. Different algorithms are used in DRL such as QL, DQNL, proximal policy optimization (PPO), deep deterministic policy gradient (DDPG), and soft actor-critic (SAC) [14].

There are different works related to RL for pendulum systems. Dao et al. [15] developed an adaptive reinforcement learning strategy with sliding mode control for unknown and disturbed wheeled inverted pendulum. Zhang et al. [16] proposed a double Q-learning reinforcement learning algorithm based on a neural network for inverted pendulum control. Baek et al. [17] developed reinforcement learning to achieve real-time control of a triple-inverted pendulum. Pal et al. [18] proposed a reinforced learning approach coupled with a proportional-integral-derivative controller for the swing up and balance of an inverted pendulum. Safeea et al. [19] established a Q-learning approach to the continuous control problem of robot inverted pendulum balancing. Lim et al. [20] established a federated reinforcement learning for controlling multiple rotary inverted pendulums in edge computing environments. Chen Ming et al. [21] developed a reinforcement learning-based control of nonlinear systems using the Lyapunov stability concept and fuzzy reward scheme for a cart pole of a pendulum system. Bi et al. [22] proposed a deep reinforcement learning approach towards the pendulum swing-up Problem based on TF agents. Manrique Escobar et al. [23] studied a deep reinforcement learning control system applied to the swing-up problem of the cart pole. Kukker and Sharma [24] proposed a genetic algorithm (GA)-optimized fuzzy Lyapunov RL controller, the controller performance is tested under different types of inverted pendulum systems. In [25] Kukker and Sharma developed stochastic genetic (SA) algorithm-assisted Fuzzy Q-Learning-based robotic manipulator control.

The RL techniques established in this work have important potential for mechatronics and robotic applications compared to the works explored in the literature and their contributions are given below:

- The developed RL methods permit the development of nonlinear controllers without understanding the physical and dynamic behavior of the system to be controlled. These RL methods are becoming more interdisciplinary, where they combine practices and theories from areas such as modern control, computational intelligence, applied optimization, and operation research to solve complex multi-objective control problems such as RIPS.
- The developed RL methods are intelligent control solutions that demonstrate computational effectiveness in solving complex control problems. These structures could alleviate the need to build heavy-duty rule-based solution mechanisms. The various intelligent approaches are customized to ensure robust swing-up control of the SLRIP and to dynamically adapt the strategies to dynamic changes in the environment such as external disturbances and unstructured dynamic variations.
- The developed RL algorithms hold a huge potential to contribute to the design and implementation of real-time controllers for automation and mechatronic systems aiming for optimized performances.
- The proposed research strategy will advance the state-of-the-art knowledge in RL with a focus on intelligent control applications. Further, this type of research can be employed by a broad range of other robotic systems.

In this paper, the swing-up and stabilization control of a SLRIP is developed by training and testing RL algorithms. The developed RL control methods are compared with the PID and LQR swing-up control. The nonlinear dynamic equations are developed using the Euler–Lagrange method. The environment of the SLRIP is MATLAB/ Simulink. Actions are actuating the horizontal arm. States are the angles and velocities of the pendulum and the horizontal arm. The reward is computed according to the angles of the pendulum and horizontal arm. The reward is zero when the pendulum attends the upright position. An agent trainer system with Q and Q network learning algorithms is proposed for the data training. The QL and DQNL control methods return more accurate results in terms of improvement compared to the PID and LQR control methods.

The rest of this paper is organized as follows. In Sect. 2, the dynamic modeling and nonlinear equations of the system are explained. In Sect. 3, the related principles of RL based on QL and DQNL algorithms for the system control are described. In Sect. 4, the simulation results are described. Finally, Sect. 5, summarizes this paper.

2 Dynamic modeling of the SLRIP

The CAD model of the SLRIP is shown in Fig. 1. The SLRIP consists of a flat horizontal arm with a pivot motion and a metal pendulum installed on its end. The pivot is fixed on the top shaft of a DC servo motor. A counterbalance mass may be fixed to the other end of the flat arm to keep the system inertia in the middle. θ_1 and θ_2 are angular positions of the 1st and 2nd links, respectively. The system has two degrees of freedom (2 DOF). The SLRIP can be modelized as a serial arm robot chain to find its kinematic model using the Denavit–Hartenberg (DH) convention. The physical parameters of the system and their values are given in Table 1. Table 2 shows the DH parameters of the system.

The homogeneous transformation matrix of the system is given in Eqs. (1) and (2). It is calculated using the parameters of DH from Table 2.

Fig. 1 CAD model of the SLRIP

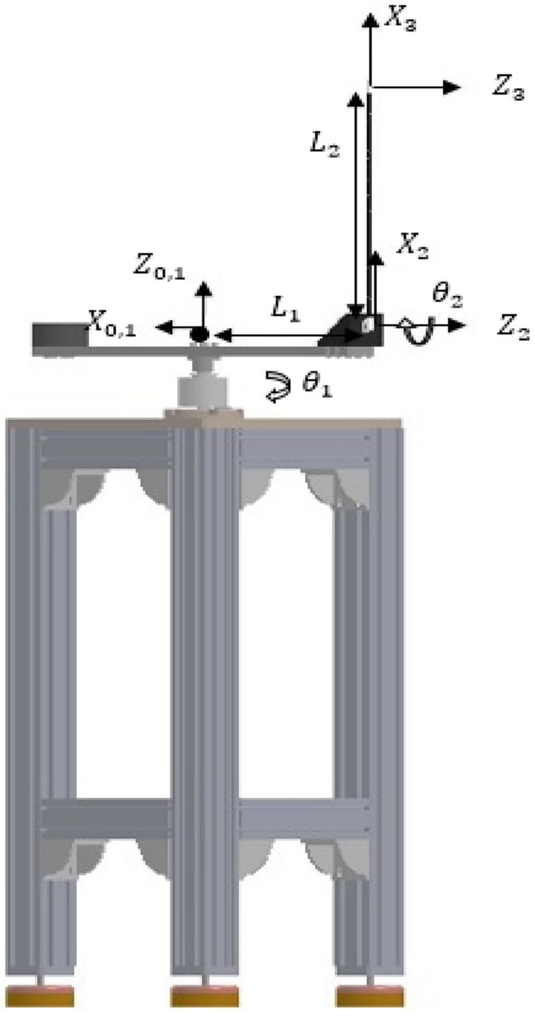

Table 1 List of parameters and their values of the SLRIP

Parameter	Description	Value
m_1	Mass of the arm	3.1129 [kg]
m_2	Mass of the pendulum	0.08 [kg]
m_B	Mass of the counterbalance	3.1469 [kg]
I_{zz1}	The inertia of the arm	0.4398 [kg m^2]
I_{zz2}	The inertia of the pendulum	0.0025 [kg m^2]
L_1	Length of the arm	0.44 [m]
L_2	Length of the pendulum	0.526 [m]
$b_{1,2}$	The viscous friction coefficient in the joints of the arm and pendulum	0.0024 [N-m-s/Rad]
g	Gravity	9.81 [N kg^{-1}]

Table 2 DH parameters

Coordinate	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	$-\frac{\pi}{2}$	0	L_1	$\theta_2 - \frac{\pi}{2}$
3	0	L_2	0	0

$$ {}^{i-1}_{i}T = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_{i-1} \\ \sin\theta_i \cos a_{i-1} & \cos\theta_i \cos a_{i-1} & -\sin a_{i-1} & -\sin a_{i-1} d_i \\ \sin\theta_i \sin a_{i-1} & \cos\theta_i \sin a_{i-1} & \cos a_{i-1} & \cos a_{i-1}\, d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1} $$

$$ {}^{0}_{3}T = {}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T \tag{2} $$

where

$$ {}^{0}_{1}T = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} {}^{1}_{2}T = \begin{bmatrix} \sin\theta_2 & \cos\theta_2 & 0 & 0 \\ 0 & 0 & 1 & L_1 \\ \cos\theta_2 & -\sin\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$
$$ {}^{2}_{3}T = \begin{bmatrix} 1 & 0 & 0 & L_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3} $$

$$ {}^{0}_{3}T = \begin{bmatrix} \sin\theta_2\cos\theta_1 & \cos\theta_2\cos\theta_1 & -\sin\theta_1 & L_2\cos\theta_1\sin\theta_2 - L_1\sin\theta_1 \\ \sin\theta_2\sin\theta_1 & \cos\theta_2\sin\theta_1 & \cos\theta_1 & L_1\cos\theta_1 + L_2\sin\theta_1\sin\theta_2 \\ \cos\theta_2 & -\sin\theta_2 & 0 & L_2\cos\theta_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4} $$

The dynamic equations of the SLRIP are found according to the kinematic model and may be given in matrix form using Eq. (5).

$$ D(\theta)\ddot{\theta} + C(\theta,\dot{\theta}) + G(\theta) = \begin{bmatrix} \tau_1 - b_1 \\ -b_2 \end{bmatrix} \tag{5} $$

where $D(\theta)$, $C(\theta,\dot{\theta})$ and $G(\theta)$ are mass matrix, Coriolis and Centripetal force vector, and gravity vector, respectively. θ, $\dot{\theta}$ and $\ddot{\theta}$ are vectors of angular positions, angular velocities, and angular accelerations, respectively. τ_1 is the torque input for the system. The mathematical model is found using the "Euler–Lagrange" method. Equation (6) is used to calculate all terms of the mass matrix.

$$ D(\theta) = \sum_{i=1}^{n} \left[(A_i)^T m_i A_i + (B_i)^T I_i B_i \right] \tag{6} $$

m_i is the mass of the system arms, $I_i \in R\ 3 \times 3$ is the inertia tensor of the system arms. A_i and $B_i \in R\ 3 \times n$ are Jacobian matrices. Furthermore, Eq. (7) is used to calculate all terms of Coriolis and Centripetal vector:

$$ C(\theta,\dot{\theta}) = \sum_{k=1}^{n}\sum_{j=1}^{n} \left[c^{i}_{kj}(\theta)\dot{\theta}_k\dot{\theta}_j \right] \tag{7} $$

$$ c^{i}_{kj}(\theta) = \frac{\partial}{\partial\theta_k} D_{ij}(\theta) - \frac{1}{2}\frac{\partial}{\partial\theta_i} D_{kj},\ 1 \le i,\ j,\ k \le n \tag{8} $$

Equation (9) is used to calculate the gravity vectors:

$$ G(\theta) = -\sum_{k=1}^{n}\sum_{j=1}^{n} \left[g_k\, m_j\, A^{j}_{ki}(\theta) \right] \tag{9} $$

The methodology to calculate the matrix elements is given as follows: Δ_{h_1} and Δ_{h_2} are gravity center vectors of the 1st and the 2nd links, respectively. The two vectors are determined corresponding to the coordinate systems of arms in Eqs. (10). Moreover, I_{m1} and I_{m2} are tensors of inertia of the 1st and the 2nd links respectively; given in Eqs. (11).

$$\Delta_{h_1} = \begin{bmatrix} 0 & \frac{L_1}{2} & 0 & 1 \end{bmatrix}^T,\ \Delta_{h_2} = \begin{bmatrix} \frac{L_2}{2} & 0 & 0 & 1 \end{bmatrix}^T \tag{10}$$

$$I_{m1} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & I_{zz_1} \end{bmatrix},\ I_{m2} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & I_{zz_2} \end{bmatrix} \tag{11}$$

The coordinates of the mass center of arms are computed corresponding to the main coordinate system.

$$h_1 = {}^0_1T\,\Delta_{h_1} = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 0 \\ \frac{L_1}{2} \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -\frac{L_1\sin\theta_1}{2} \\ \frac{L_1\cos\theta_1}{2} \\ 0 \\ 1 \end{bmatrix} \tag{12}$$

$$h_2 = {}^0_2T\,\Delta_{h_2} = \begin{bmatrix} \frac{L_2\cos\theta_1\sin\theta_2}{2} - L_1\sin\theta_1 \\ \frac{L_2\sin\theta_1\sin\theta_2}{2} + L_1\cos\theta_1 \\ \frac{L_2\cos\theta_2}{2} \\ 1 \end{bmatrix} \tag{13}$$

where

$${}^0_2T = \begin{bmatrix} \sin\theta_2\cos\theta_1 & \cos\theta_2\cos\theta_1 & -\sin\theta_1 & -L_1\sin\theta \\ \sin\theta_2\sin\theta_1 & \cos\theta_2\sin\theta_1 & \cos\theta_1 & L_1\cos\theta_1 \\ \cos\theta_2 & -\sin\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{14}$$

The Jacobian matrix of the 1st arm is obtained by the calculation of the derivative of the vector h_1 according to θ_1 and θ_2. ξ_i denotes the type of joint, in this case $\xi_i = 1$ because it's a rotary motion. 'i' is a unit vector of the 3rd column of the coordinate system. Furthermore, the variables z^1 and ξ_1 are used. The first arm is a rotational link $\xi_1 = 1$ and $b_1 = \xi_1 z^1 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$. The Jacobian matrix of the 1st arm is calculated below:

$$z^1 = {}^0_1R\,i^2 = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{15}$$

$$J_1 = \begin{bmatrix} \frac{\partial}{\partial\theta_1}\left(-\frac{L_1\sin\theta_1}{2}\right) & \frac{\partial}{\partial\theta_2}\left(-\frac{L_1\sin\theta_1}{2}\right) \\ \frac{\partial}{\partial\theta_1}\left(\frac{L_1\cos\theta_1}{2}\right) & \frac{\partial}{\partial\theta_2}\left(\frac{L_1\cos\theta_1}{2}\right) \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} -\frac{L_1\cos\theta_1}{2} & 0 \\ -\frac{L_1\sin\theta_1}{2} & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} \tag{16}$$

J_1 of the 1st arm can be divided into two matrices A_1 and B_1:

$$A_1 = \begin{bmatrix} -\frac{L_1 \cos\theta_1}{2} & 0 \\ -\frac{L_1 \sin\theta_1}{2} & 0 \\ 0 & 0 \end{bmatrix}, B_1 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} \tag{17}$$

The Jacobian matrix of the 2nd arm is calculated using the derivative of the vector h_2 according to θ_1 and θ_2. The 2nd arm is rotational which means $\xi_2 = 1$ and $b_2 = \xi_2 z^2 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$. The Jacobian matrix of the 2nd arm is calculated below:

$$z^2 = {}^0_2R\, i^2 = \begin{bmatrix} \cos\theta_1 \sin\theta_2 & \cos\theta_1 \cos\theta_2 & -\sin\theta_1 \\ \sin\theta_1 \sin\theta_2 & a\cos\theta_2 \sin\theta_1 & \cos\theta_1 \\ \cos\theta_2 & -\sin\theta_2 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -\sin\theta_1 \\ \cos\theta_1 \\ 0 \end{bmatrix} \tag{18}$$

$$J_2 = \begin{bmatrix} \frac{\partial}{\partial\theta_1}\left(\frac{L_2 \cos\theta_1 \sin\theta_2}{2} - L_1 \sin\theta_1\right) & \frac{\partial}{\partial\theta_2}\left(\frac{L_2 \cos\theta_1 \sin\theta_2}{2} - L_1 \sin\theta_1\right) \\ \frac{\partial}{\partial\theta_1}\left(\frac{L_2 \sin\theta_1 \sin\theta_2}{2} + L_1 \cos\theta_1\right) & \frac{\partial}{\partial\theta_2}\left(\frac{L_2 \sin\theta_1 \sin\theta_2}{2} + L_1 \cos\theta_1\right) \\ \frac{\partial}{\partial\theta_1}\left(\frac{L_2 \cos\theta_2}{2}\right) & \frac{\partial}{\partial\theta_2}\left(\frac{L_2 \cos\theta_2}{2}\right) \\ 0 & -\sin\theta_1 \\ 0 & \cos\theta_1 \\ 1 & 0 \end{bmatrix} \tag{19}$$

J_2 of the 2nd arm is divided into two matrices A_2 and B_2:

$$A_2 = \begin{bmatrix} -\frac{L_2 \sin\theta_1 \sin\theta_2}{2} - L_1 \cos\theta_1 & \frac{L_2 \cos\theta_1 \cos\theta_2}{2} \\ \frac{L_2 \cos\theta_1 \sin\theta_2}{2} - L_1 \sin\theta_1 & \frac{L_2 \cos\theta_2 \cos\theta_1}{2} \\ 0 & \frac{-L_2 \sin\theta_2}{2} \end{bmatrix} B_2 = \begin{bmatrix} 0 & -\sin\theta_1 \\ 0 & \cos\theta_1 \\ 1 & 0 \end{bmatrix} \tag{20}$$

The mass matrices of the two arms are given in Eqs. (21) and (22), respectively.

$$D(\theta_1) = m_1 A_1^T A_1 + B_1^T I_1 B_1 = \begin{bmatrix} \frac{m_1 L_1^2}{4} + I_{zz_1} & 0 \\ 0 & 0 \end{bmatrix} \tag{21}$$

$$D(\theta_2) = m_2 A_2^T A_2 + B_2^T I_2 B_2 = \begin{bmatrix} m_2\left(\frac{L_2^2 \sin^2\theta_2}{4} + L_1^2\right) & -\left(\frac{L_1 L_2 m_2 \cos\theta_2}{2}\right) \\ -\left(\frac{L_1 L_2 m_2 \cos\theta_2}{2}\right) & \left(\frac{L_2^2 m_2}{4} + I_{zz_2}\right) \end{bmatrix} \tag{22}$$

The inertial tensors of arms are calculated based on the main coordinate system:

$$I_1 = {}^0_1R\, I_{m1}\, {}^0_1R^T,\ I_2 = {}^0_2R\, I_{m2}\, {}^0_2R^T \tag{23}$$

$D(\theta)$ matrix of the system is provided in Eq. (24):

$$D(\theta) = D(\theta_1) + D(\theta_2) = \begin{bmatrix} \frac{m_1 L_1^2}{4} + I_{zz_1} + m_2\left(\frac{L_2^2 \sin^2\theta_2}{4} + L_1^2\right) & -\left(\frac{L_1 L_2 m_2 \cos\theta_2}{2}\right) \\ -\left(\frac{L_1 L_2 m_2 \cos\theta_2}{2}\right) & \left(\frac{L_2^2 m_2}{4} + I_{zz_2}\right) \end{bmatrix} \tag{24}$$

The elements of the velocity coupling matrix of the two arms are found below:

$$\begin{aligned} C_1 &= \begin{bmatrix} \frac{1}{2}\frac{\partial}{\partial\theta_1} D_{11}(\theta) & \frac{1}{2}\frac{\partial}{\partial\theta_1} D_{12}(\theta) \\ \frac{\partial}{\partial\theta_2} D_{11}(\theta) - \frac{1}{2}\frac{\partial}{\partial\theta_1} D_{21}(\theta) & \frac{\partial}{\partial\theta_2} D_{12}(\theta) - \frac{1}{2}\frac{\partial}{\partial\theta_1} D_{22}(\theta) \end{bmatrix} \\ &= \begin{bmatrix} 0 & 0 \\ \frac{1}{4}\left(L_2^2 m_2 \sin(2\theta_2)\right) & \frac{1}{2}\left(L_1 L_2 m_2 \sin(\theta_2)\right) \end{bmatrix} \end{aligned} \tag{25}$$

$$C_2 = \begin{bmatrix} \frac{\partial}{\partial\theta_1} D_{21}(\theta) - \frac{1}{2}\frac{\partial}{\partial\theta_2} D_{11}(\theta) & \frac{\partial}{\partial\theta_1} D_{22}(\theta) - \frac{1}{2}\frac{\partial}{\partial\theta_2} D_{12}(\theta) \\ \frac{1}{2}\frac{\partial}{\partial\theta_2} D_{21}(\theta) & \frac{1}{2}\frac{\partial}{\partial\theta_2} D_{22}(\theta) \end{bmatrix} = \begin{bmatrix} -\frac{1}{8}\left(L_2^2 m_2 \sin(2\theta_2)\right) & -\frac{1}{4}\left(L_1 L_2 m_2 \sin(\theta_2)\right) \\ \frac{1}{4}\left(L_1 L_2 m_2 \sin(\theta_2)\right) & 0 \end{bmatrix} \tag{26}$$

The $C(\theta,\dot{\theta})$ vector is found by the verification of the elements' equality of the matrices given below.

$$\begin{bmatrix} 0 & 0 \\ \frac{1}{4}\left(L_2^2 m_2 \sin(2\theta_2)\right) & \frac{1}{2}\left(L_1 L_2 m_2 \sin(\theta_2)\right) \end{bmatrix} = \begin{bmatrix} \dot{\theta}_1^2 & \dot{\theta}_1\dot{\theta}_2 \\ \dot{\theta}_1\dot{\theta}_2 & \dot{\theta}_2^2 \end{bmatrix} \tag{27}$$

$$\begin{bmatrix} -\frac{1}{8}\left(L_2^2 m_2 \sin(2\theta_2)\right) & -\frac{1}{4}\left(L_1 L_2 m_2 \sin(\theta_2)\right) \\ \frac{1}{4}\left(L_1 L_2 m_2 \sin(\theta_2)\right) & 0 \end{bmatrix} = \begin{bmatrix} \dot{\theta}_1^2 & \dot{\theta}_1\dot{\theta}_2 \\ \dot{\theta}_1\dot{\theta}_2 & \dot{\theta}_2^2 \end{bmatrix} \tag{28}$$

The $C(\theta,\dot{\theta})$ vector of the system is provided below:

$$C(\theta,\dot{\theta}) = \begin{bmatrix} \frac{1}{2}\left(L_1 L_2 m_2 \dot{\theta}_2^2 \sin(\theta_2)\right) + \frac{1}{4}\left(L_2^2 m_2 \dot{\theta}_1 \dot{\theta}_2 \sin(2\theta_2)\right) \\ -\frac{1}{8}\left(L_2^2 \dot{\theta}_1^2\, m_2 \sin(2\theta_2)\right) \end{bmatrix} \tag{29}$$

The gravity vector of the SLRIP is:

$$G = \begin{bmatrix} 0 \\ -\frac{1}{2}\left(L_2 g\, m_2 \sin(\theta_2)\right) \end{bmatrix} \tag{30}$$

The matrix form of the nonlinear equations of the system is given below:

$$\begin{bmatrix} \frac{m_1 L_1^2}{4} + I_{zz_1} + m_2\left(\frac{L_2^2 \sin^2\theta_2}{4} + L_1^2\right) & -\left(\frac{L_1 L_2 m_2 \cos\theta_2}{2}\right) \\ -\left(\frac{L_1 L_2 m_2 \cos\theta_2}{2}\right) & \left(\frac{L_2^2 m_2}{4} + I_{zz_2}\right) \end{bmatrix}\begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix} + \begin{bmatrix} \frac{1}{2}\left(L_1 L_2 m_2 \dot{\theta}_2^2 \sin(\theta_2)\right) + \frac{1}{4}\left(L_2^2 m_2 \dot{\theta}_1 \dot{\theta}_2 \sin(2\theta_2)\right) \\ -\frac{1}{8}\left(L_2^2 \dot{\theta}_1^2\, m_2 \sin(2\theta_2)\right) \end{bmatrix} + \begin{bmatrix} 0 \\ -\frac{1}{2}\left(L_2 g\, m_2 \sin(\theta_2)\right) \end{bmatrix} = \begin{bmatrix} \tau_1 - b_1 \\ - b_2 \end{bmatrix}(31) \tag{31}$$

The control input of the system is the torque τ_1 of the servo motor. The applied torque is described by Eq. (32). This torque equation is implemented in the MATLAB/Simscape model.

$$\tau_1 = \frac{\eta_m k_t}{R_m} V_m - \frac{\eta_m k_t k_m}{R_m}\dot{\theta} \tag{32}$$

where V_m is the motor input voltage, k_t is the motor torque constant, k_m is the motor speed constant, η_m is motor efficiency coefficient, R_m is terminal resistance and $\dot{\theta}$ is the angular velocity. A CAD dynamic model of the SLRIP was established via MATLAB/Simscape Toolbox to prove the mathematical model. MATLAB/Simscape and CAD models of the SLRIP are shown in Fig. 2a, b, respectively. To simulate the two models, the initial conditions of the angular positions of arms are chosen as $\theta_1 = 0°$, and $\theta_2 = 20°$. Furthermore, the same results were obtained by two models (Fig. 3).

3 Deep reinforcement learning control for SLRIP

3.1 Q-learning

DRL is a combination of DL structure and RL idea, but it focuses more on RL. RL solves decision problems and makes RL technology practically and successfully. Furthermore, RL is a kind of machine learning based on that the system

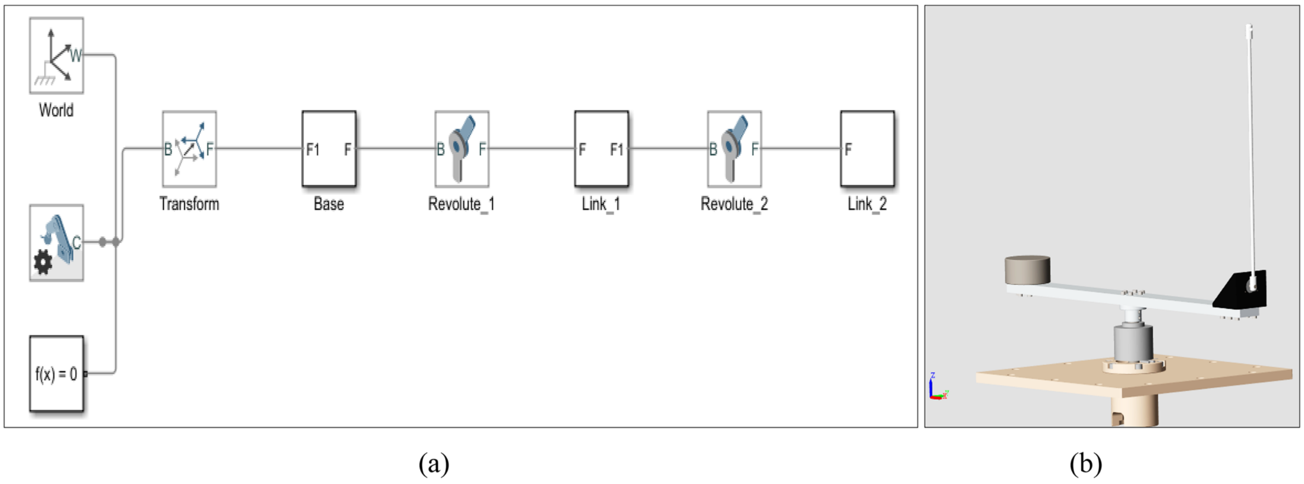

Fig. 2 SLRIP: (a) Simscape model (b) CAD model in Matlab/Simulink

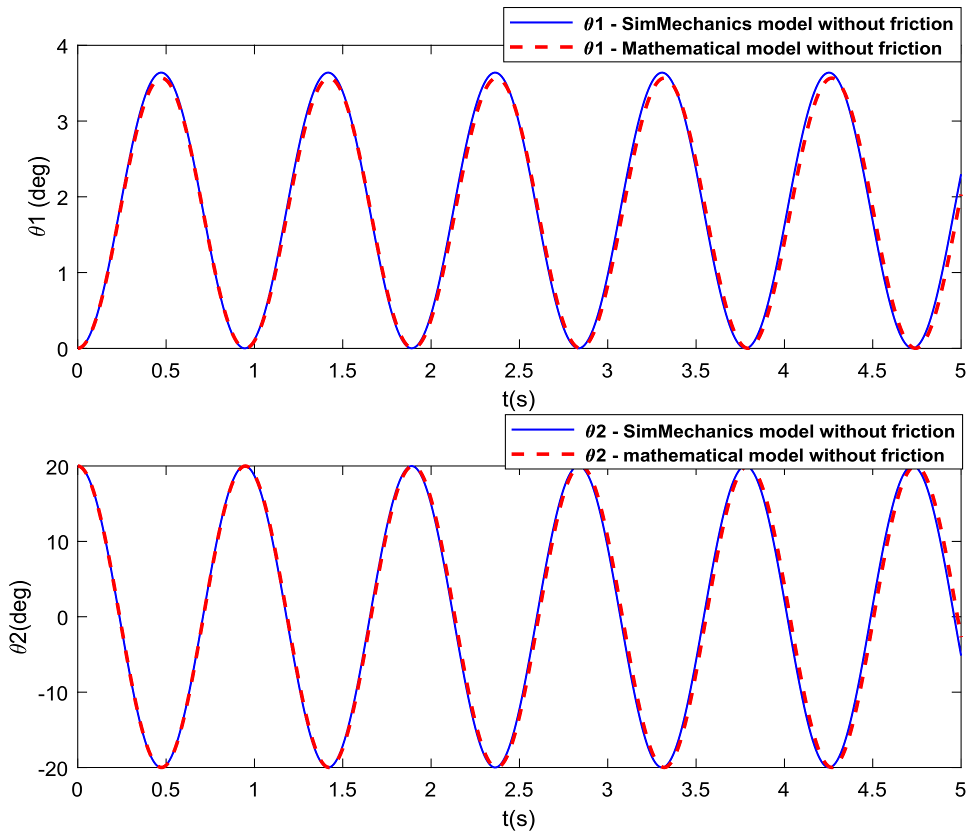

Fig. 3 Comparison of θ_1 and θ_2 obtained from mathematical and Simscape models of the SLRIP

can learn from its environment by training and error minimization. The two main parts of RL are the environment and the agent. The agent is a decision-maker, and the environment defines everything outside the agent. The environment uses the agent input (a_t) to create the state (s_t) and reward $\left(r_t\right)$ which are used as inputs for the agent as well. RL algorithm is required for the agent to calculate the optimal actions according to the policy (π) in order to attain the maximum value of the reward. RL is a closed-loop system with has a loop connection between the agent and the

Table 3 Q-Table

States	Actions			
	a_1	a_2	…	a_m
s_1	$Q(s_1, a)$	$Q(s_1, a_2)$	…	$Q(s_1, a_m)$
s_2	$Q(s_2, a_1)$	$Q(s_2, a)$	…	$Q(s_2, a_m)$
…			…	…
s_n	$Q(s_n, a_1)$	$Q(s_n, a_2)$	…	$Q(s_n, a_m)$

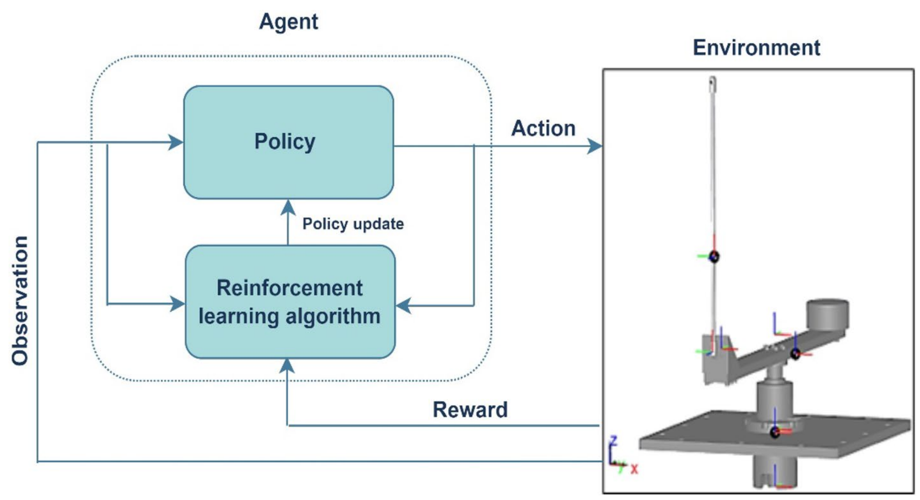

Fig. 4 Block diagram of the Q-Learning control for SLRIP

environment. A loop can be named a step and an episode has N steps. The process of training is accomplished based on the step numbers (N_s) and episode numbers (N_e), where ($N_s \times N_e$) is the loop time. The connection between the state and the action is linked using a table called Q or Q-table as policy [26]. Each value in the table is noted as $Q(s_i, a_j)$ where $i = (1, 2, \dots n), j = (1, 2, \dots m)$. Table 3 illustrates an example of Q-Table. The block diagram of the Q-learning for SLRIP is shown in Fig. 4. The action is calculated using Eq. (33).

$$a_t = \begin{cases} maxQ_t(s_i, a_j) - with\,probability(1 - \varepsilon) \\ anyaction(a_i, a_j) - with\,probability\underline{\varepsilon} \end{cases} \tag{33}$$

where ε is a value for any random action $\varepsilon \in (0, 1]$. The RL objective is to determine the optimal policy that has a maximum reward. The Q-Table is used to find the best action for each state in the environment. A Bellman equation is used at each state to get the estimated future state and reward that will be compared with other states.

$$Q_{new}(s_t, a_t) \leftarrow r_{t+1} + \gamma(\max Q(s_{t+1}, a_{t+1})) \tag{34}$$

where the updater or new Q-value is $Q_{new}(s_t, a_t)$. γ is a discount factor and demonstrates the importance of future rewards to the current state with $\gamma \in (0, 1]$. r_{t+1} is the reward received by the agent at time $t + 1$ while performing an action (a) to move from one state to another. $\max Q(s_{t+1}, a_{t+1})$ is the maximum Q-value of the state s_{t+1} with action a_{t+1} at time $t + 1$. The agent generates the new action a_t according to the current state s_t using Eq. (33), it will give order to the environment to do the new state and reward s_{t+1} and r_{t+1} respectively. The state s_{t+1} is used to find $\max Q(s_{t+1}, a_{t+1})$ in all Q-values related to actions in Table 3. Hence, the $Q_{new}(s_t, a_t)$ is calculated and updated in the Q-Table. Different applications need a high number of states and actions, which makes the Q-table highly large and increases the calculation time of all Q-values. Thus, DQNL is used to avoid Q-Table restrictions. The working of the Q-learning algorithm is given below:

Q-learning algorithm

```
Input data: γ discount factor; ε ∈ (0,1]
Algorithm initialization
Initialize $Q\ (s_t, a_t)$ randomly, except $Q$ (terminal)
$Q$ (terminal) $\leftarrow 0$;
Calculation of Q-value and updating the Q-Table for each step in each episode
for each episode do
    Initialize state S;
    for each step-in episode do
        do
            $A \leftarrow Select\ action\ (Q, S, \varepsilon)$;
            $Q_{new}\ (s_t, a_t) \leftarrow r_{t+1} + \gamma\ (\max Q\ (s_{t+1}, a_{t+1}))$
            $S \leftarrow S'$
        while S isn't terminal;
    end
end
```

3.2 Deep Q-network learning

The DQN approximates the Q-value by using a NN model as an alternative to the Q-table of the previous method. The NN creates a prediction model with the input layer which is the environment' states; the hidden layers, and the output layer is the predicted Q-value. Furthermore, an extra NN model is used as a target model to estimate the max $Q\left(s_{t+1}, a_{t+1}\right)$ and to calculate the $Q_{new}\left(s_t, a_t\right)$[27].The error between $Q_{new}\left(s_t, a_t\right)$ and $Q_{new\_NN}\left(s_t, a_t\right)$ will be used to update the weight of the prediction model by applying the gradient descent method.

The loss function is given below:

$$Loss = \left[Q_{new}\left(s_t, a_t\right) - Q_{new\_NN}\left(s_t, a_t\right)\right]^2 \tag{35}$$

A set of data is a requirement for the training process. Arbitrary rewards and states are created and need to be sent to the RL agent. The prediction model is applied to predict the $Q_{new\_NN}\left(s_t, a_t\right)$ by using the policy of Eq. (33) to choose the optimal action for the environment. Moreover, the environment is in a closed loop with the new reward and state. The set of training data might be given in Eq. (36).

$$E_t = \left(s_t, a_t, r_{t+1}, a_{t+1}\right) \tag{36}$$

where the E_t is saved in an experience reply loop applied in the training step. One random sample E_t is chosen for target and prediction models with s_t and s_{t+1} as inputs, respectively. The block diagram of the deep Q-network learning control for SLRIP is shown in Fig. 5. Figure 6 shows the flow chart of the DQNL algorithm.

4 Simulation results and discussion

4.1 Proposed methods

In this work, the SLRIP is controlled by training and testing RL algorithms, see Fig. 7. Positions and velocities obtained from 1st and 2nd links are inputs of the RL model. Furthermore, the torque input to the system is the output of the RL model. The control of the SLRIP system is based on two processes: the swing-up of and stabilization of the pendulum. The agents are developed to compute the swing angle of the pendulum until it achieves the upright position. When the pendulum is near the upright position (180° ± 10°) the proximal policy agent switches the stability action. A deep NN agent was implemented for the control of the DQNL. The parameters of the DQNL agent are presented in Table 4. The DQNL agent with 1 input layer,

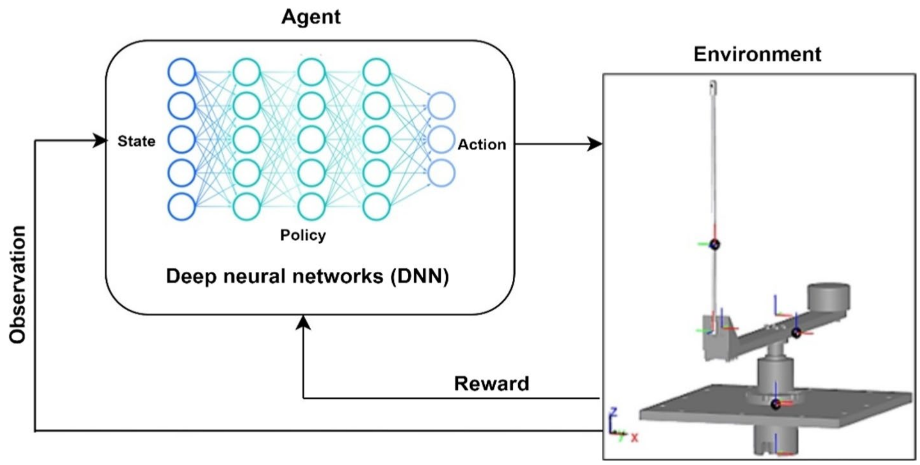

Fig. 5 Block diagram of the Deep Q-network learning control for SLRIP

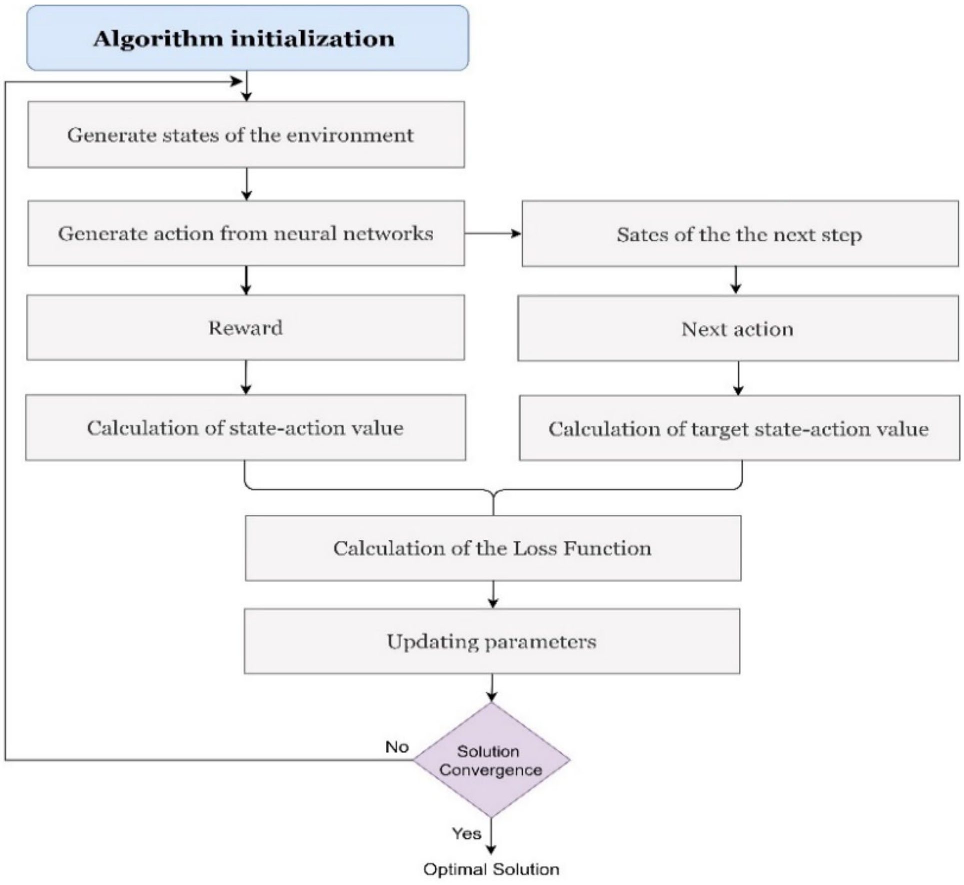
Algorithm initialization
Generate states of the environment
Generate action from neural networks
Sates of the the next step
Reward
Next action
Calculation of state-action value
Calculation of target state-action value
Calculation of the Loss Function
Updating parameters
No
Solution Convergence
Yes
Optimal Solution

Fig. 6 Flow chart of the DQNL algorithm

25 hidden layers, and 1 output layer is proposed for the swing-up control of the SLRIP. The RL environment is the Simscape model of the SLRIP. The observation vector of RL is $[\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2]$. The reward (*r*) signal of the RL is given in the equation below:

$$r = -\theta_2^2 - 0.2(\theta_1 - \theta_2)^2 - 0.15\dot{\theta}_1 + T \tag{37}$$

where T is the torque input:

$$T = \left\{ \begin{array}{ll} 35 & if\theta_2 \in (180°, \pm 10°]and\theta_1 \pm 180° \\ 0 & \text{otherwise} \end{array} \right\} \tag{38}$$

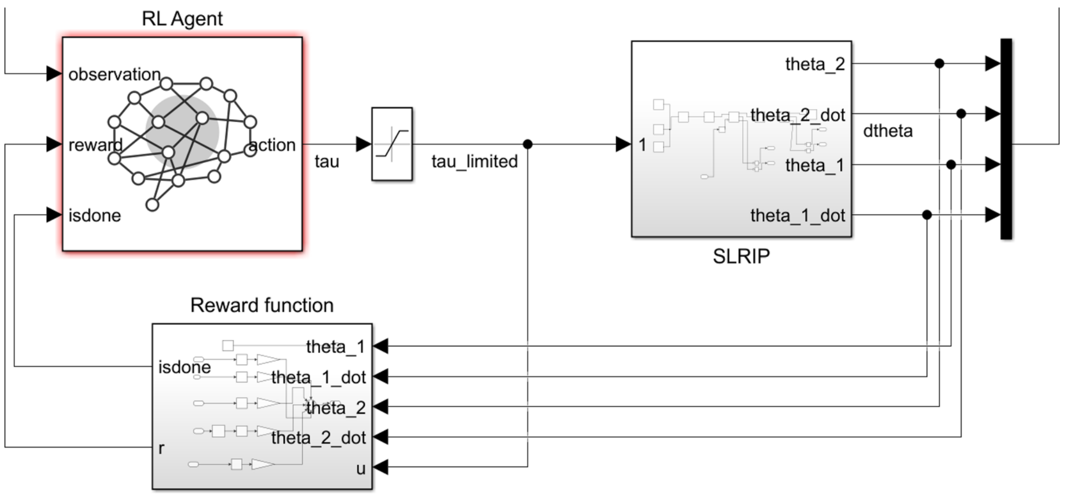

Fig. 7 Simulink model of the DQNL control of the SLRIP

Table 4 DQNL agent parameters

Parameter	Value
Environment	SLRIP
Reward	$r = -\theta_2^2 - 0.2\left(\theta_1 - \theta_2\right)^2 - 0.15\dot{\theta}_1 + T$ $T = \left\{ \begin{array}{ll} 35 & \textit{if}\ \theta_2 \in (180°, \pm 10°]\ \textit{and}\ \theta_1 \pm 180° \\ 0 & \text{otherwise} \end{array} \right\}$
Action	$\theta_1 = \pm 180°$

Fig. 8 Training performance of the agent of the SLRIP

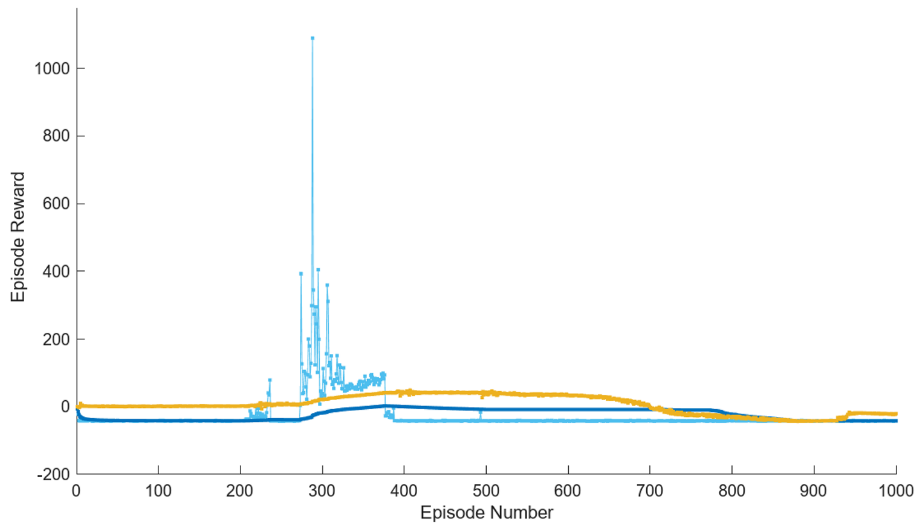

Figure 7 shows the Simulink model of the DQNL control of the SLRIP. The training performance of the agent is shown in Fig. 8.

The complexity of the DQNL algorithm is related to numerous parameters such as the dimensions of ANN, the number of neurons, episodes, iterations, and even the size of the actions and states of the environment. The parameters of the simulation and RL algorithms are described in Table 5.

As mentioned in Table 5, the agent's inputs are the environment observations vector, reward signal from the environment, and Isdone, a flag to terminate episode simulation. The agent's output is the action (control signal of the system). The parameters of the RL algorithm are the sample rate is 4s, the discount factor is 0.99 and the learning rate is 0.005.

Table 5 Parameters of the simulation and RL algorithm

Parameters of the simulation and RL algorithms	
Agent algorithm	DQNL
Environment	SLRIP
Number of the input layer	1
Number of hidden layers	25
Number of the output layer	1
Observation	$[\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2]$
Action	τ_1
Max episodes	1000
Max episode length	500
Time of the feedback controller	0.001
Simulation time	20 s
Sample rate	4 s
Learning rate	0.005
Discount factor	0.99

The frequency at which the agent receives feedback from the environment is known as the sample rate, while the extent to which the agent updates its internal model based on the newly received information is determined by the learning rate. The MATLAB software is utilized to conduct the simulation experiment. 1000 episodes are chosen for the agent to train the SLRIP model. The mean reward per episode is around 175. The training concluded once all agents reached the stopping criteria for training. The simulation is carried out on a PC with the following configuration: CPU (11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz 1.69 GHz) and RAM (16 GB). The QL algorithm demonstrates convergence after a 12-h training process, whereas a mere 0.70 h suffice to train the DQNL algorithm, thereby reducing the training time by a significant 94.16%.

The design of QL and DNQL controllers can pose several challenges in simulation, one of the main difficulties is finding a balance between exploration and exploitation. The controller needs to explore the environment to find optimal policies. The determination of an appropriate exploration strategy is an important task, as random exploration can be inefficient, and systematic exploration can lead to sub-optimal performance. The QL and DNQL can struggle with problems that have a large state or action space. As the number of possible states or actions increases, the computation and memory requirements also increase exponentially. The training of QL and DNQL controllers is sensitive to hyperparameter settings and initial conditions, and they struggle to converge to an optimal policy. The learning process can be unstable, with the Q-values oscillating or diverging. Furthermore, ensuring the convergence and stability of the learning algorithm requires careful consideration of learning rates, discount factors, and other parameters. In environments where rewards are sparse or delayed, it can be challenging for Q-learning and DNQL agents to learn effective policies. Without frequent rewards, it may take a considerable amount of time for the agent to learn desired behaviors or to explore the state space adequately. Moreover, in environments where rewards are sparse or delayed, it can be challenging for QL and DNQL agents to learn effective policies. Without frequent rewards, it may take a considerable amount of time for the agent to learn desired behaviors or to explore the state space adequately.

4.2 Results and discussion

The simulation results and discussion are presented in this section. The DQNL algorithm is established for the swing-up control of the SLRIP. The agent should keep the 2nd link in the upright stability position by supplying the system with a suitable torque input. DQNL algorithm should find the optimal input torque value that is needed to balance the pendulum. Moreover, the DQNL algorithm could balance the pendulum and make it stable with high efficiency. Additionally, the swing-up control of SLRIP using the DRL algorithms was tested with the conventional LQR and PID controllers to validate its effectiveness. The simulation results shown in Figs. 9, 10, and 11 indicate that all controllers have succeeded in controlling the SLRIP. The performance of controllers was studied based on the comparison of errors such as integral time absolute error (ITAE), integral time square error (ITSE), integral absolute error (IAE), and integral square error (ISE) and root mean square error (RMSE), which are given in Tables 6, 7, and 8.

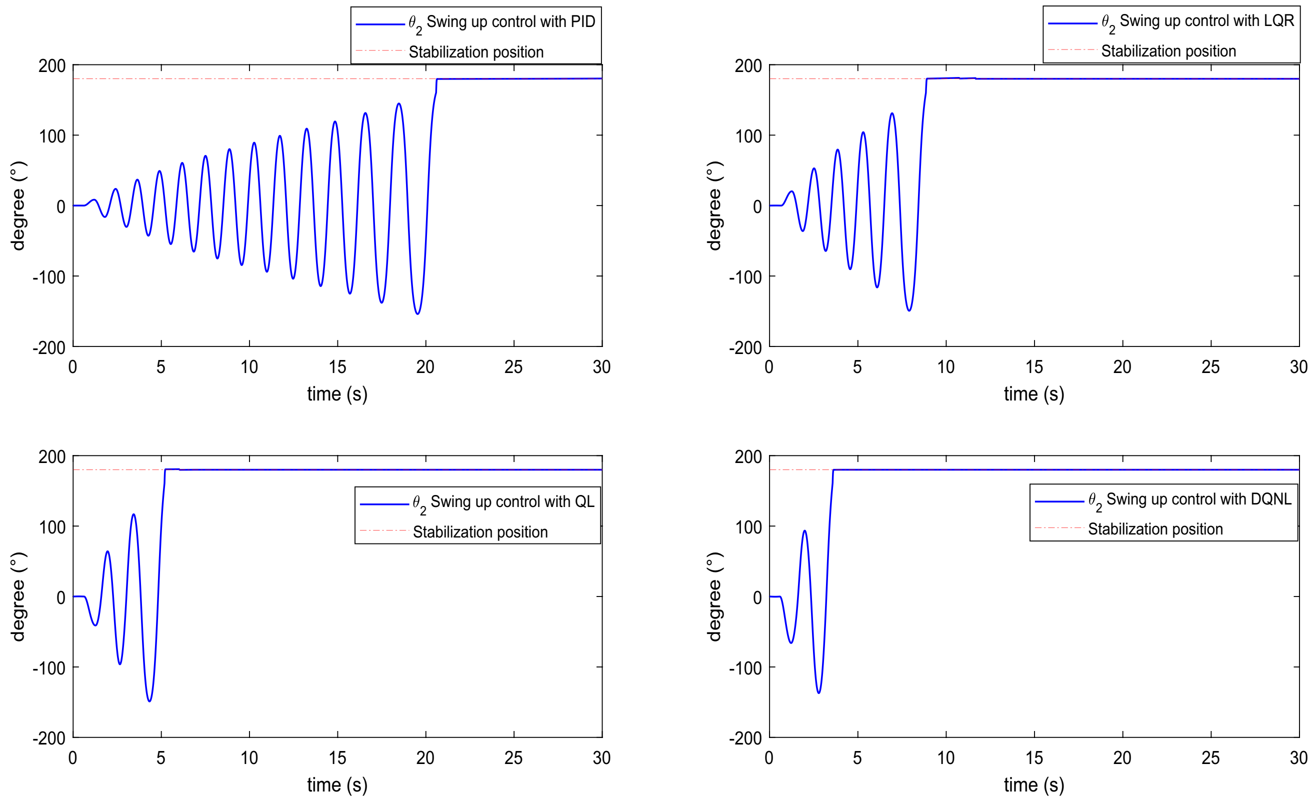

Fig. 9 Angle position θ_2 of the swing-up control of the pendulum link

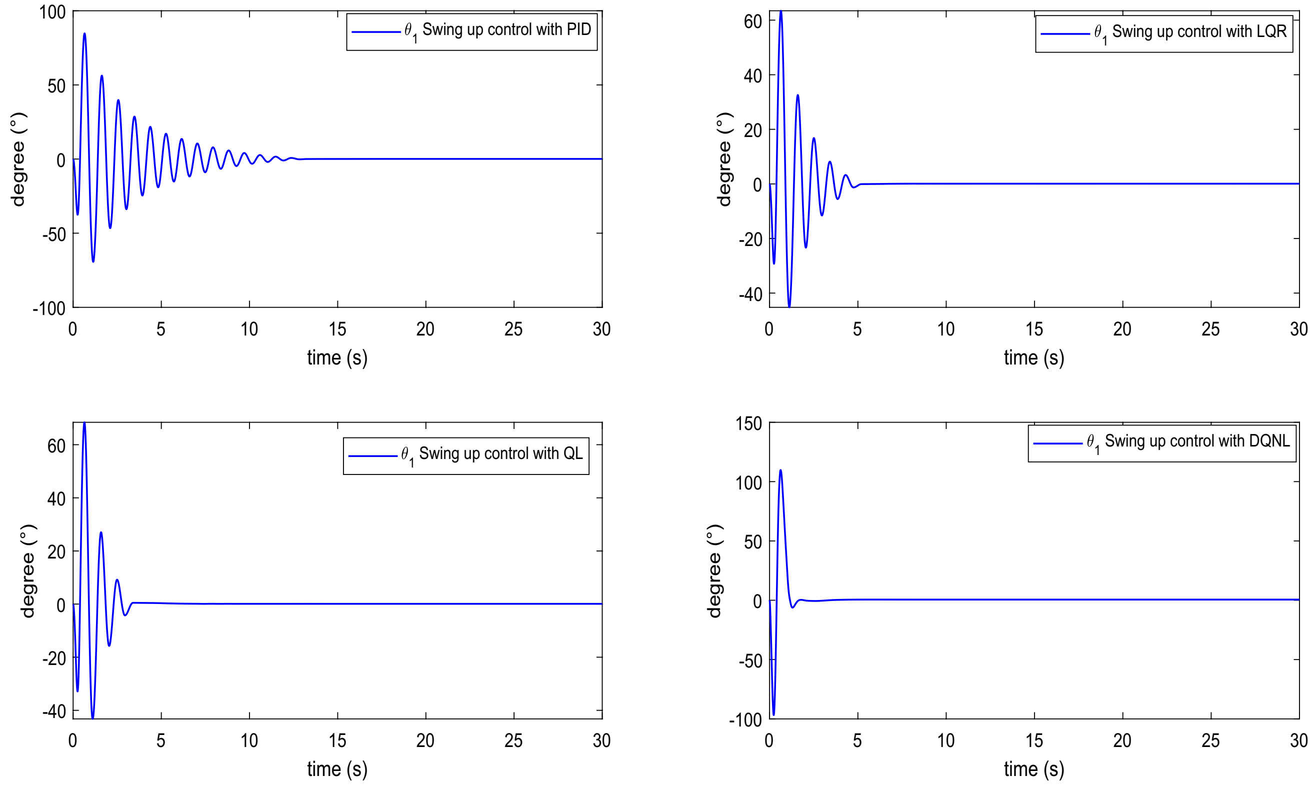

Fig. 10 Angle position θ_1 of the swing-up control of the horizontal link

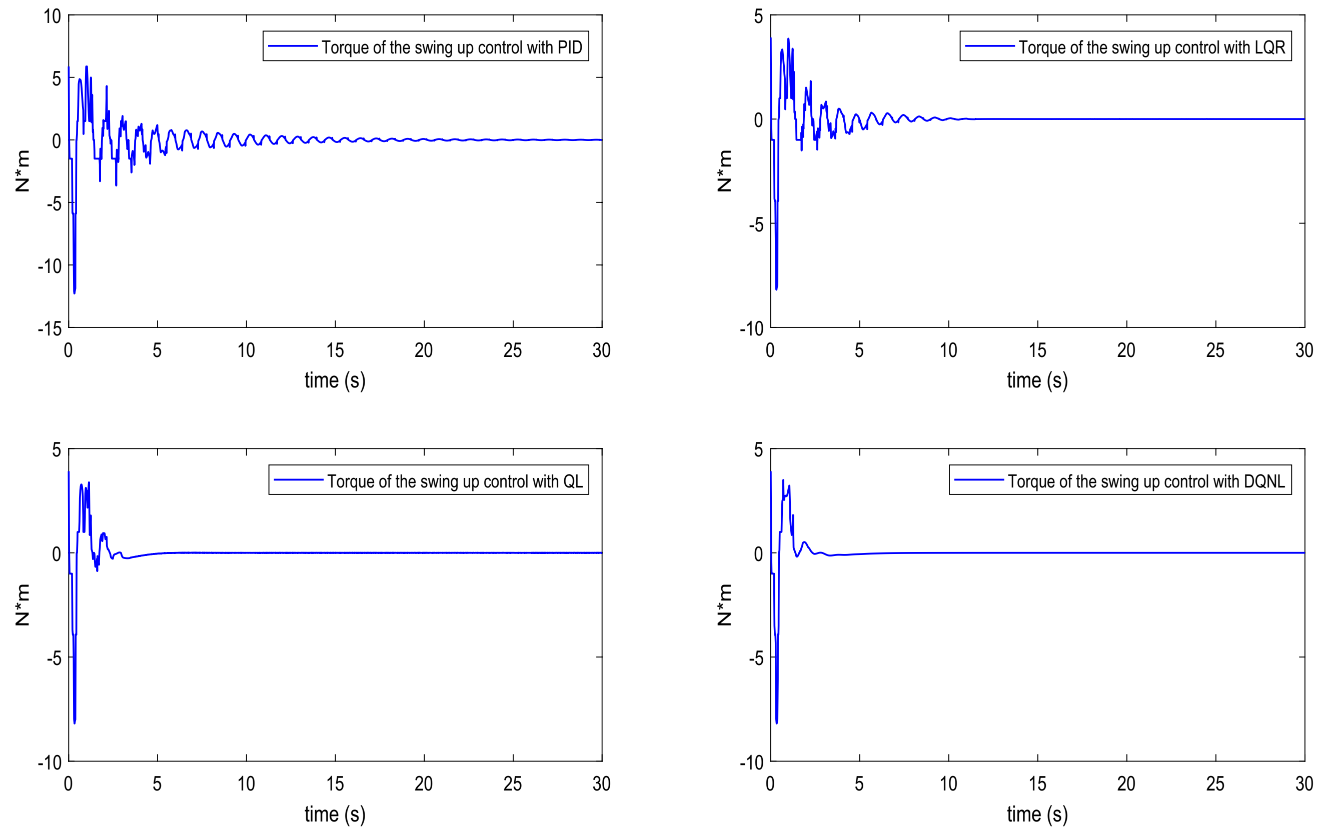

Fig. 11 The input control signal of the swing-up controllers

Table 6 Comparison of θ_2 for swing-up controllers in terms of IAE, ISE, ITAE, ITSE and RMSE

Errors	Controllers			
	PID	LQR	QL	DQNL
IAE	3.7184×10^{-3}	1.6103×10^{-3}	978.5668	683.1916
ISE	7.7546×10^{-5}	3.3673×10^{-5}	2.1160×10^{-5}	1.4793×10^{-5}
ITAE	1.6738×10^{-8}	7.2489×10^{-7}	4.4050×10^{-7}	3.0754×10^{-7}
ITSE	3.4907×10^{-10}	1.5158×10^{-10}	9.5254×10^{-9}	6.6589×10^{-9}
RMSE	160.76	105.95	84.0	70.24

Table 7 Comparison of θ_1 for swing-up controllers in terms of IAE, ISE, ITAE, ITSE and RMSE

Errors	Controllers			
	PID	LQR	QL	DQNL
E	172.3459	72.4547	62.1360	11.9867
ISE	5.9986×10^{-3}	2.1460×10^{-3}	2.0577×10^{-3}	108.5353
ITAE	7.7582×10^{-6}	3.2615×10^{-6}	2.7970×10^{-6}	5.3958×10^{-5}
ITSE	2.7003×10^{-8}	9.6602×10^{-7}	9.2628×10^{-7}	4.8857×10^{-6}
RMSE	14.13	8.45	8.28	1.901

According to the results presented in Table 6, it can be inferred that the RMSE serves as a critical parameter for evaluating the effectiveness of the controllers. The QL algorithm yields significantly more precise control outcomes, demonstrating an improvement percentage of 47.74% and 20.71% in terms of RMSE relative to the PID and LQR controllers, respectively. Moreover, the DQNL algorithm yields significantly more precise control outcomes, demonstrating an improvement percentage of 56.30% and 33.70% in terms of RMSE relative to the PID and LQR controllers, respectively.

Table 8 Comparison of the input control signal for swing-up controllers in terms of IAE, ISE, ITAE, ITSE, and RMSE

Errors	Controllers			
	PID	LQR	QL	DQNL
IAE	12.5831	5.9901	4.0818	3.7589
ISE	37.1928	15.1676	13.4198	13.1669
ITAE	5.6626×10^{-6}	2.6956×10^{-6}	1.8369×10^{-6}	1.6916×10^{-6}
ITSE	1.6737×10^{-7}	6.8257×10^{-6}	6.0391×10^{-6}	5.9253×10^{-6}
RMSE	1.11	0.71	0.669	0.6627

IAE, ISE, ITAE, and ITSE have been computed as popular performance indexes. However, their performance is not the same. Accordingly, an optimization algorithm can be incorporated into QL and DQNL algorithms to adjust the IAE, ISE, ITAE, and ITSE for both RL controllers.

Based on the results presented in Table 7, it can be inferred that the RMSE serves as a critical parameter for evaluating the effectiveness of the controllers. The QL algorithm yields significantly more precise control outcomes, demonstrating an improvement percentage of 41.40% and 2.01% in terms of RMSE relative to the PID and LQR controllers, respectively. Moreover, the DQNL algorithm yields significantly more precise control outcomes, demonstrating an improvement percentage of 48.97% and 14.67% in terms of RMSE relative to the PID and LQR controllers, respectively. IAE, ISE, ITAE, and ITSE have been computed as popular performance indexes. However, their performance is not the same. Accordingly, an optimization algorithm can be incorporated into QL and DQNL algorithms to adjust the IAE, ISE, ITAE, and ITSE for both RL controllers.

According to the results presented in Table 8, it can be inferred that the RMSE serves as a critical parameter for evaluating the effectiveness of the controllers. The QL algorithm yields significantly more precise control outcomes, demonstrating an improvement percentage of 39.72% and 5.77% in terms of RMSE relative to the PID and LQR controllers, respectively. Moreover, the DQNL algorithm yields significantly more precise control outcomes, demonstrating an improvement percentage of 40.29% and 6.66% in terms of RMSE relative to the PID and LQR controllers, respectively. IAE, ISE, ITAE, and ITSE have been computed as popular performance indexes. However, their performance is not the same. Accordingly, an optimization algorithm can be incorporated into QL and DQNL algorithms to adjust the IAE, ISE, ITAE, and ITSE for both RL controllers. Table 9 presents a summary of the improvement percentage in terms of RMSE between RL and classical controllers.

In summary, The SLRIP is nonlinear and inherently unstable. It requires precise and adaptive control to maintain stability and balance. Traditional control methods developed in this work (PID, LQR) might struggle to handle the complexities and dynamic changes of the system. QL and DQNL are model-free techniques that do not require explicit knowledge of the system dynamics. They can adaptively learn the control policy by interacting with the system and observing its responses, making them suitable for handling nonlinearity and instability. Furthermore, SLRIP requires continuous control actions, such as applying torque to the 1st arm to adjust the pendulum's position. Both QL and DQNL are extended to handle continuous control problems by using function approximation techniques. These techniques enable the agent to learn a continuous control policy that can effectively balance the pendulum on top of the rotating base. In this work, the developed QL and DQNL are relevant for controlling a SLRIP due to their ability to handle nonlinearities, instability, high-dimensional state spaces, and continuous control actions. These algorithms provide an effective framework for training an agent to learn an optimal control policy in such complex and dynamic systems. Based on the findings derived from our analysis and considering the latest scholarly publications in the field, the results indicate great promise. It is essential to establish a double deep Q-network learning algorithm in order to thoroughly compare its performance with other

Table 9 Comparison of swing-up controllers in terms of the relative improvements' percentages of RMSE

RMSE	Controllers			
	PID versus QL (%)	LQR versus QL (%)	PID versus DQNL (%)	LQR versus DQNL (%)
Pendulum link	47.74	20.71	56.30	33.70
Horizontal link	41.40	2.01	48.97	14.67
Input control signal	39.72	5.77	40.29	6.66

classical controllers, such as the controller proposed by Y Dai et al. in their work [28]. Furthermore, it is crucial to conduct tests with alternative RL agents like the soft actor-critic–proximal policy optimization (SAC–PPO) technique presented in [29]. To ensure the reliability and accuracy of the RL algorithms, it is imperative to utilize real experimental setups, similar to the one employed by D. Brown et al. [30]. Additionally, to further enhance the control outcomes achieved by the RL controllers, it is recommend the development of an adaptive reinforcement learning strategy combined with sliding mode control, as demonstrated in [15].

5 Conclusion

In this paper, the RL algorithms for the swing-up and stabilization control of a SLRIP were developed in MATLAB simulation. The control approach consists of the environment, which is the Simscape model of the SLRIP, and the agent is the controller. QL and DQNL are both model-free control algorithms used without a profound background of understanding of classical control theory. The QL and DQNL agents were developed for the swing-up and stabilization control action of the SLRIP. 1000 episodes were required for the training of the agent. Furthermore, the learning algorithm is sensitive to the selection of parameters such as learning rates or discount factors and these parameters may be effective in the control of SLRIP. According to the obtained results, the QL control method returns more accurate results in terms of improvement percentage from 39.72 to 47.74% and from 2.01 to 20.71% compared to the PID and LQR control methods; respectively. Moreover, the DQNL control method returns more accurate results in terms of improvement percentage from 40.29 to 56.30% and from 6.66 to 33.70% compared to the PID and LQR control methods; respectively. For future works, RL controllers will be validated under a real experimental setup of a SLRIP. Moreover, the developed algorithm will be used to control more complex environments such as double and triple-link rotary inverted pendulum systems to prove the efficiency of the control algorithm.

Author contributions Contribution of paper: RL control algorithms are developed for the swing-up and stabilization control of a single-link rotary inverted pendulum (SLRIP) and compared with classic control methods. -The RL algorithms are used without a deep understanding of the classical controllers. The outcome indicates the effectiveness of the QL and DQNL algorithms compared to the conventional controllers. This work presents effective machine-learning approaches in the field of control engineering.

Funding The publication of this work is supported by the University of Technology Bahrain (UTB) under Grant EX610610.

Data availability The data of this study are available upon request from the corresponding author.

Declarations

Competing interests The author declares no competing interests.

References

1. Hazem ZB, Bingül Z. Comprehensive review of different pendulum structures in engineering applications. IEEE Access. 2023. https://doi.org/10.1109/ACCESS.2023.3269580.
2. Kumar G, Kumar R, Kumar A. A review of the controllers for structural control. Arch Comput Methods Eng. 2023. https://doi.org/10.1007/s11831-023-09931-y.
3. Önen Ü. Model-free controller design for nonlinear underactuated systems with uncertainties and disturbances by using extended state observer based chattering-free sliding mode control. IEEE Access. 2023;11:2875–85. https://doi.org/10.1109/ACCESS.2023.3234864.
4. Hou M, Zhang X, Chen D, Xu Z. Hierarchical sliding mode control combined with nonlinear disturbance observer for wheeled inverted pendulum robot trajectory tracking. Appl Sci. 2023;13(7):4350. https://doi.org/10.3390/app13074350.

5. Jingwen H, et al. Control of rotary inverted pendulum using model-free backstepping technique. IEEE Access. 2019;7:96965–73. https://doi.org/10.1109/ACCESS.2019.2930220.
6. Saifizul AA, Zainon Z, Abu Osman NA, Azlan CA, Ibrahim UU. Intelligent control for self-erecting inverted pendulum via adaptive neuro-fuzzy inference system. Am J Appl Sci. 2006;3(4):1795–802.
7. Saleem O, Abbas F, Iqbal J. Complex fractional-order LQIR for inverted-pendulum-type robotic mechanisms: design and experimental validation. Mathematics. 2023;11(4):913. https://doi.org/10.3390/math11040913.
8. Alatabani LE, Ali ES, Saeed RA. Machine learning and deep learning approaches for robotics applications. In: Artificial intelligence for robotics and autonomous systems applications. Cham: Springer; 2023. p. 303–33. https://doi.org/10.1007/978-3-031-28715-2_10.
9. Hazem ZB, Bingül Z. A comparative study of anti-swing radial basis neural-fuzzy LQR controller for multi-degree-of-freedom rotary pendulum systems. Neural Comput Appl. 2023. https://doi.org/10.1007/s00521-023-08599-6.
10. Wai RJ, Chang LJ. Stabilizing and tracking control of nonlinear dual-axis inverted-pendulum system using fuzzy neural network. IEEE Trans Fuzzy Syst. 2006;14(1):145–68. https://doi.org/10.1109/TFUZZ.2005.859305.
11. Singh B, Kumar R, Singh VP. Reinforcement learning in robotic applications: a comprehensive survey. Artif Intell Rev. 2022. https://doi.org/10.1007/s10462-021-09997-9.
12. Kukker A, Sharma R. Neural reinforcement learning classifier for elbow, finger and hand movements. J Intell Fuzzy Syst. 2018;35(5):5111–21. https://doi.org/10.3233/JIFS-169795.
13. Israilov S, Fu L, Sánchez-Rodríguez J, Fusco F, Allibert G, Raufaste C, Argentina M. Reinforcement learning approach to control an inverted pendulum: A general framework for educational purposes. PLoS ONE. 2023;18(2):e0280071. https://doi.org/10.1371/journal.pone.0280071.
14. Din AFU, Mir I, Gul F, Akhtar S. Development of reinforced learning based non-linear controller for unmanned aerial vehicle. J Amb Intell Humaniz Comput. 2023;14(4):4005–22. https://doi.org/10.1007/s12652-022-04467-8.
15. Dao PN, Liu YC. Adaptive reinforcement learning strategy with sliding mode control for unknown and disturbed wheeled inverted pendulum. Int J Control Autom Syst. 2021;19(2):1139–50. https://doi.org/10.1007/s12555-019-0912-9.
16. Zhang D, Wang X, Li X, Wang D. Inverted pendulum control of double q-learning reinforcement learning algorithm based on neural network. Sci Bull Ser D Mech Eng. 2020;82(2):15–26.
17. Baek J, Lee C, Lee YS, Jeon S, Han S. Reinforcement learning to achieve real-time control of triple inverted pendulum. Eng Appl Artif Intell. 2024;128: 107518. https://doi.org/10.1016/j.engappai.2023.107518.
18. Pal AK, Nestorović T. Swing up and balance of an inverted pendulum using reinforced learning approach coupled with a proportional-integral-derivative controller. In: International conference on electrical, computer, communications and mechatronics engineering (ICECCME), IEEE, pp. 1–6, 2022. https://doi.org/10.1109/ICECCME55909.2022.9988506.
19. Safeea M, Neto P. A Q-learning approach to the continuous control problem of robot inverted pendulum balancing. Intell Syst Appl. 2023. https://doi.org/10.1016/j.iswa.2023.200313.
20. Lim HK, Kim JB, Kim CM, Hwang GY, Choi HB, Han YH. Federated reinforcement learning for controlling multiple rotary inverted pendulums in edge computing environments. In: International conference on artificial intelligence in information and communication (ICAIIC), IEEE, pp. 463–464, 2020. https://doi.org/10.1109/ICAIIC48513.2020.9065233
21. Chen M, Lam HK, Shi Q, Xiao B. Reinforcement learning-based control of nonlinear systems using Lyapunov stability concept and fuzzy reward scheme. IEEE Trans Circuits Syst II Express Briefs. 2019;67(10):2059–63. https://doi.org/10.1109/TCSII.2019.2947682.
22. Bi Y, Chen X, Xiao C. A Deep reinforcement learning approach towards pendulum swing-up problem based on TF-Agents. arXiv preprint arXiv:2106.09556, 2021.
23. Manrique Escobar CA, Pappalardo CM, Guida D. A parametric study of a deep reinforcement learning control system applied to the swing-up problem of the cart-pole. Appl Sci. 2020;10(24):9013. https://doi.org/10.3390/app10249013.
24. Kukker A, Sharma R. Genetic algorithm-optimized fuzzy lyapunov reinforcement learning for nonlinear systems. Arab J Sci Eng. 2020;45(3):1629–38. https://doi.org/10.1007/s13369-019-04126-9.
25. Kukker A, Sharma R. Stochastic genetic algorithm-assisted fuzzy q-learning for robotic manipulators. Arab J Sci Eng. 2021;46(10):9527–39. https://doi.org/10.1007/s13369-021-05379-z.
26. Xi A, Chen C. Walking control of a biped robot on static and rotating platforms based on hybrid reinforcement learning. IEEE Access. 2019;8:148411–24. https://doi.org/10.1109/ACCESS.2020.3015506.
27. Dang KN, Van LV. Development of deep reinforcement learning for inverted pendulum. Int J Electr Comput Eng. 2023;13(4):2088–8708. https://doi.org/10.11591/ijece.v13i4.pp3895-3902.
28. Dai Y, Lee K, Lee S. A real-time HIL control system on rotary inverted pendulum hardware platform based on double deep Q-network. Measur Control. 2021;54(3–4):417–28. https://doi.org/10.1177/00202940211000380.
29. Bhourji RS, Mozaffari S, Alirezaee S. Reinforcement learning ddpg–ppo agent-based control system for rotary inverted pendulum. Arab J Sci Eng. 2023. https://doi.org/10.1007/s13369-023-07934-2.
30. Brown D, Strube M. Design of a neural controller using reinforcement learning to control a rotational inverted pendulum. In: International conference on research and education in mechatronics (REM), IEEE, pp. 1–5, 2020. https://doi.org/10.1109/REM49740.2020.9313887.