CAAI Transactions on Intelligence Technology

Research Article

Adaptive PID controller based on *Q*-learning algorithm

ISSN 2468-2322
Received on 29th June 2018
Revised on 27th September 2018
Accepted on 8th October 2018
doi: 10.1049/trit.2018.1007
www.ietdl.org

Qian Shi[1], *Hak-Keung Lam*[1] ✉, *Bo Xiao*[2], *Shun-Hung Tsai*[3]

[1]*Department of Informatics, King's College London, Bush House, Strand Campus, 30 Aldwych, London WC2B 4BG, UK*
[2]*Hamlyn Centre for Robotic Surgery, Imperial College London, London SW7 2AZ, UK*
[3]*Graduate Institute of Automation Technology, National Taipei University of Technology, Taipei 10608, Taiwan*
✉ *E-mail: hak-keung.lam@kcl.ac.uk*

Abstract: An adaptive proportional–integral–derivative (PID) controller based on *Q*-learning algorithm is proposed to balance the cart–pole system in simulation environment. This controller was trained using *Q*-learning algorithm and implemented the learned *Q*-tables to change the gains of linear PID controllers according to the state of the system during the control process. The adaptive PID controller based on *Q*-learning algorithm was trained from a set of fixed initial positions and was able to balance the system starting from a series of initial positions that are different from the ones used in the training session, which achieved equivalent or even better performances in comparison with the conventional PID controller and the controller only uses *Q*-learning algorithm. This indicates the advantage of the adaptive PID controller based on *Q*-learning algorithm both in the generality of balancing the cart–pole system from a relatively wide range of initial positions and in the stabilisability of achieving smaller steady-state error.

1 Introduction

Various controllers have been developed and widely used in industrial environments such as proportional–integral–derivative (PID) controllers, fuzzy logic-based controllers because of the high robustness and scalability with the simplicity in structure. Nevertheless, these conventional controllers suffer from the high dependency on parameters. These controllers usually are not optimally tuned and able to reach satisfactory performances in real applications and require frequent adjustments when damages occur or being in adverse situations [1]. Therefore, the tuning parameters of conventional controllers to achieve an optimal performance become an outstanding issue [2]. Additionally, these controllers have less satisfactory performance in complex environment, especially in the situations associated with practical applications considering the high non-linearity and uncertainty of the environment. These drawbacks address the importance of developing alternative control techniques. Contrarily, reinforcement learning (RL) algorithm is a machine-learning algorithm that generates optimal policy by interacting with environment and receiving reinforcement signals under the circumstance where the dynamics and the underlying of the environment remain unknown. This property proposes a promising solution to solving the systems with high non-linearity by avoiding formulating accurate mathematical models [3]. Combining conventional control methods with RL is proved to be very powerful and efficient in solving practical problems like robotic control problems in high-dimensional state and action spaces [4]. There are several successful applications that have been implemented such as robot navigation [3], autonomous helicopter control [5] etc., showing a promising prospect of employing RL algorithms in control systems.

Great amounts of researches have applied the RL algorithm to achieve innovative control strategies. One of them is to totally replace the conventional controllers by the RL algorithms. In [6], five different RL algorithms (two value-iteration algorithms and three policy-iteration algorithms) were tested to replace a double-loop PID controller to control a ball screw feed drive, and all the algorithms indicated the superiority of the RL algorithms to the conventional PID controller. Ramanathan *et al.* [7] used Q-learning algorithm to control the level of a liquid in a non-linear conical tank and a satisfactory performance was achieved. Although ideal achievements were observed in the applications stated above, it is also noticeable that since in RL algorithms the choosing of state representation, output and reward all have intensive effects on the performance of the controller, designing controller by implementing RL algorithms individually becomes more challenging when the dimensions of the state and action space increase [2]. Another considerable approach is to combine the RL algorithm with the conventional controllers, where the RL algorithms are implemented to address the tuning issue that the conventional controllers have.

To solve this tuning issue, several systematic methods have been proposed in [8]. Ziegler–Nichols methods [9] proposed in 1995 is regarded as a classical tuning rule, which has a broad implementation in industrial environment. Genetic algorithms were applied to provide an optimal conventional controller in the research described in [10]. Besides, a novel approach applied to optimise the parameters for non-linear PID parameters called ant colony optimisation algorithm was proposed and implemented in [11, 12]. Both methods showed better performance compared with conventional controllers. However, most of these approaches have high requirements on computational ability, while RL algorithms significantly reduce the learning time [8].

Regarding the RL-based controllers, several techniques have been proposed in previous researches. A methodology named Discrete Action Reinforcement Learning Automata was proposed in [13], two extension versions of this method were proposed in [14, 15], aiming to tune the parameters of one PID controller with higher learning speed. Both controllers were tested on an Automatic Voltage Regulator system. Similarly, Continuous Action Reinforcement Learning automata for PID controller [13] was applied as an on-line tuning method in the research done by Howell and Best [16] and Mohammadi *et al.* [17]. In the research of [17], the approach was also tested via an engine idle-speed control system and the Ball and Beam System in simulation environment. Both showed superior performance compared with the conventional controllers tuned following the Ziegler–Nichols methods. However, the integral and interpolation operation in the algorithm brings the extra cost of computation. Another method named actor–critic method used to tune PID parameters was implemented in [18] as to implement wind turbine control, and a similar method was implemented in [1] as to achieve tracking function in a special class of linear systems. In both algorithms, the actor mapped states to PID controller parameters and the critic provided the evaluation of the output of actor. Both approaches indicated better performances contrasted to the conventional PID controllers. Meanwhile, *Q*-learning algorithm [19] is also widely used in tuning parameters for controllers. Researches [8, 20] used this algorithm to tune fuzzy controllers. Carlucho *et al.* [2] tuned the PID controller using an incremental Q-PID algorithm by dynamically dividing the actions into more specific areas to obtain higher controlling accuracy. The simplicity in utilisation and independency to the system parameters make *Q*-learning algorithm more reliable in real applications [14]. Nevertheless, less attention was paid to implementing *Q*-learning algorithm to tune parameters for multiple PID controllers. One related research is [21], which tuned two sets of PID controllers to control the angular and linear speed of a soccer robot and achieved good performance in speed response and stability. However, in this research the learning process of finding optimal parameters for PID controller required long computation period, and less analyses were conducted on the learned controller regarding the stability or robustness.

As to address the open issue of implementing *Q*-learning algorithm on multiple PID controllers, an adaptive PID controller based on *Q*-learning algorithm is proposed in this research. It adapts to the similar approach implemented in [21] by varying the values of gains of linear PID controllers according to different operating space of system state after training with *Q*-learning algorithm, instead of having a set of fixed gains through the whole controlling progress. Generally, this innovative controller implements direct control through the PID controllers in the lower level while achieving adaptive adjustment of controller gains by learning through *Q*-learning algorithm in higher level, which improves the performance of the controllers in non-linear and complex systems. Additionally, in order to have a more comprehensive analysis of the approach, the proposed controller was tested on the classical cart–pole problem in simulation environment in comparison with conventional PID controllers and the controller only using *Q*-learning algorithm. The main contributions of this paper are summarised as follows:

(i) An innovative structure of controller is proposed. The controller consists of two linear PID controllers with *Q*-learning algorithm applied to adaptively change the values of gains during the controlling process.
(ii) The performance of the controller is tested on the cart–pole system in simulation environment. Both the learning efficiency and the states of the cart–pole system are analysed.
(iii) Two other categories of controllers are provided as comparison, which are traditional linear PID controller and the controller only applies *Q*-learning algorithm. The comparison indicates the benefits of the adaptive PID controller in generality and stability.

The structure of rest of the paper is organised as follows. In Section 2, a brief introduction of relevant theories is conducted. The methodology and technical details are described in Section 3 and the simulation results of three different controllers are illustrated and compared in Section 4. Finally in Section 5, a conclusion is drawn and the future work is mentioned.

2 Preliminary

2.1 Linear PID controller

Fig. 1 illustrates the structure of a typical linear PID controller, which consists of three components, namely proportional, integral and derivative part. Equation (1) provides the output of the controller

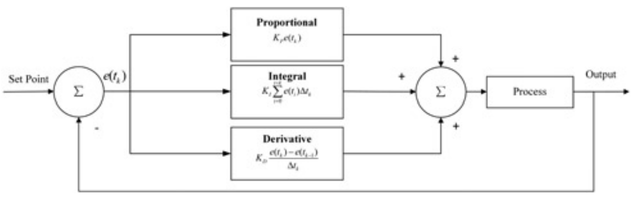

Fig. 1 *Structure of PID controller*

in discrete time form:

$$u(t_k) = K_{\mathrm{P}} e(t_k) + K_{\mathrm{I}} \sum_{i=0}^{k} e(t_i)\Delta t_k + K_{\mathrm{D}} \frac{e(t_k) - e(t_{k-1})}{\Delta t_k} \qquad (1)$$

where t_k is the kth time step; $u\big(t_k\big)$ is the output in kth time step; $e\big(t_k\big)$ is the error in kth time step; Δt_k is the interval of sampling time in simulation; and $K_{\mathrm{P}}, K_{\mathrm{I}}, K_{\mathrm{D}}$ are the proportional, integral and derivative gains, respectively.

2.2 Reinforcement learning

RL is a goal-directed problem-solving approach which belongs to one of the approaches in machine learning. This approach was inspired by the nature of learning, which learns to generate control policy only by interacting with environment without knowing the underlying of the system model. The RL problem can be defined as agent–environment interaction shown in Fig. 2. In this interaction, the agent observes states $S_t \in \mathcal{S}$ and takes actions $A_t \in \mathcal{A}$. The environment receives actions, updates new states $S_{t+1} \in \mathcal{S}$ and outputs a scalar reward $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$. The agent then receives the reward, stays in new states $S_{t+1} \in \mathcal{S}$ and choses new actions $A_{t+1} \in \mathcal{A}$. This interaction continues until reaching the terminal state $S_{\mathrm{T}} \in \mathcal{S}$. The goal of the agent is to derive an optimal control policy which maximises the discounted accumulated rewards, named as expected discounted return G_t [as defined in (2)], in the long term:

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}. \qquad (2)$$

where γ is the discounting factor, $0 < \gamma < 1$.

The RL problem is based on Markov decision processes (MDP) [22]. An MDP has the property that the current state encodes enough information for the agent to make decision despite the previous states it has visited. There are two main categories of RL algorithms, which are policy-based ones and value-based ones. In value-based RL, a value function is used to estimate the value of being in each state. The definition of the state-value function from

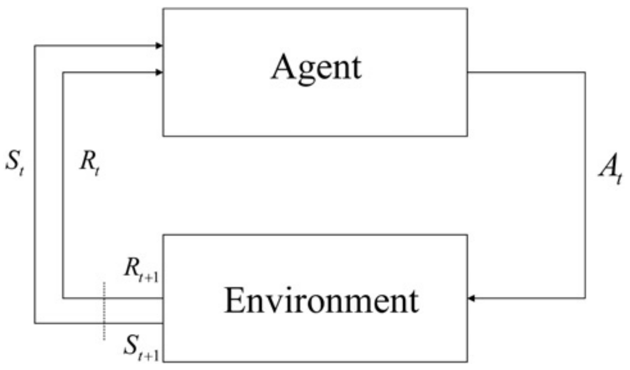

Fig. 2 *Illustration of agent–environment interaction in RL*

a given state under policy π is

$$v_{\pi}(s) \doteq \mathbb{E}_{\pi}\left[G_t | S_t = s\right] = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s\right], \tag{3}$$

where $\mathbb{E}_{\pi}$ denotes the expectation under policy π.

Similarly, the action-value function q_{π} of taking action a in state s under policy π can be defined as

$$\begin{aligned} q_{\pi}(s, a) &\doteq \mathbb{E}_{\pi}[G_t | S_t = s, A_t = a] \\ &= \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a\right]. \end{aligned} \tag{4}$$

Among all the action-value functions, the optimal action-value function, defined as

$$q_{*}(s, a) \doteq \max_{\pi} q_{\pi}(s, a), \tag{5}$$

generates the optimal policy π_{*}.

The Q-learning algorithm [23] is an off-policy value-based learning algorithm of RL. In this algorithm, the learned action-value function, Q, directly approximates the optimal action-value function, q^{*}, instead of following the current policy. The updating rule is

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_{a} Q(S_{t+1}, a) - Q(S_t, A_t)], \tag{6}$$

where α denotes the learning rate.

The general procedures of the Q-learning algorithm is presented in Algorithm 1 (see Fig. 3) [22].

2.3 Cart–pole problem

The standard cart–pole problem is shown in Fig. 4. A rigid pendulum is connected to a cart with a hinge. The pendulum is able to rotate in the vertical plane, while the cart is restricted to move in the one-dimension track. A horizontal force $u(t)$ is applied to this system as to balance the pendulum. There are four variables used to describe the state of the cart–pole system, which are $\theta(t)$, $\dot{\theta}(t)$, $x(t)$ and $\dot{x}(t)$ denoting the angle of the pendulum, angular velocity, displacement of the cart and the velocity of the cart, respectively.

```
Initialize $Q(s, a), \forall a \in \mathcal{A}, \forall s \in \mathcal{S}$, arbitrarily;
For every episode;
repeat
    Initialize state $S_t$;
    repeat
        Perform action $A_t$, observe $S_{t+1}$ and $R_{t+1}$;
        Update $Q$ values according to Eq. (6);
        $S_t \leftarrow S_{t+1}$
    until $S_t$ reaches terminal state $S_T$;
until episode ends;
```

Fig. 3 *Algorithm 1: Q-learning algorithm*

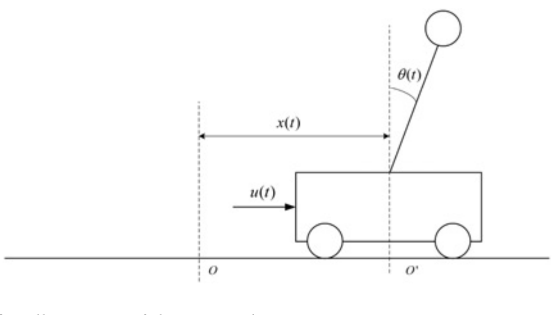

Fig. 4 *Illustration of the cart–pole system*

The dynamic model [24] is given below:

$$\dot{x}_1(t) = x_2(t) \tag{7}$$

$$\dot{x}_2(t) = \frac{\begin{pmatrix} -F_1(M+m)x_2(t) - m^2 l^2 x_2(t)^2 \sin x_1(t) \cos x_1(t) \\ +F_0 m l x_4(t) \cos x_1(t) + (M+m) m g l \sin x_1(t) \\ -m l \cos x_1(t) u(t) \end{pmatrix}}{(M+m)(J+ml^2) - m^2 l^2 (\cos x_1(t))^2} \tag{8}$$

$$\dot{x}_3(t) = x_4(t) \tag{9}$$

$$\dot{x}_4(t) = \frac{\begin{pmatrix} F_1 m l x_2(t) \cos x_1(t) + (J+ml^2) m l x_2(t)^2 \sin x_1(t) \\ -F_0((J+ml^2) x_4(t) - m^2 g l^2 \sin x_1(t) \cos x_1(t) \\ +(J+ml^2) u(t) \end{pmatrix}}{(M+m)(J+ml^2) - m^2 l^2 (\cos x_1(t))^2}, \tag{10}$$

where $x_1(t)$ is the displacement of angle (rad); $x_2(t)$ is the angular velocity (rad/s); $x_3(t)$ is the position of the cart (m); $x_4(t)$ is the linear velocity of the cart (m/s); $u(t)$ is the force applied to the cart (N); M is the mass of the cart (kg); m is the mass of the pendulum (kg); l is half length of the pendulum (m); J is the moment of inertia of the pendulum, $J = ml^2/3\,\text{kg}\,\text{m}^2$; F_0 is the friction factor of the cart (N/m/s); F_1 is the friction factor of the pendulum (N/rad/s); and g is the gravity acceleration, $g = 9.8\,\text{m/s}^2$.

3 Methodology

In this section, the details of the methodology are introduced. The whole architecture of the adaptive PID controller based on Q-learning algorithm is introduced in Section 3.1. The learning process using Q-learning algorithm is explained in Section 3.2.

3.1 Design of the controller

The adaptive PID controller based on Q-learning algorithm proposed was designed to balance the cart–pole system. The architecture of the controller is shown in Fig. 5. The direct control of the system is accomplished by two linear PID controllers, while the adaption of the parameters is based on the learned Q-tables, which are obtained from the training process using Q-learning algorithm.

As shown in Fig. 5, the control of the pendulum and the cart is implemented by PID controllers. At time step t_k, PID controller 1 takes $\theta_{\text{ref}} = 0$ as reference, inputs the error of the angle $e_p(t_k)$ defined as

$$e_p(t_k) = \theta_{\text{ref}} - \theta(t_k) \tag{11}$$

and outputs the force $u_p(t_k)$, which is defined as

$$\begin{aligned} u_p(t_k) &= K_{\text{PP}} e_p(t_k) + K_{\text{IP}} \sum_{i=0}^{k} e_p(t_i) \Delta t_k \\ &+ K_{\text{DP}} \frac{e_p(t_k) - e_p(t_{k-1})}{\Delta t_k}, \end{aligned} \tag{12}$$

where K_{PP}, K_{IP} and K_{DP} are the proportional, integral and derivative gain of PID controller 1, respectively.

PID controller 2 takes $x_{\text{ref}} = 0$ as reference, inputs the error of the position $e_c(t_k)$ defined as

$$e_c(t_k) = x_{\text{ref}} - x(t_k) \tag{13}$$

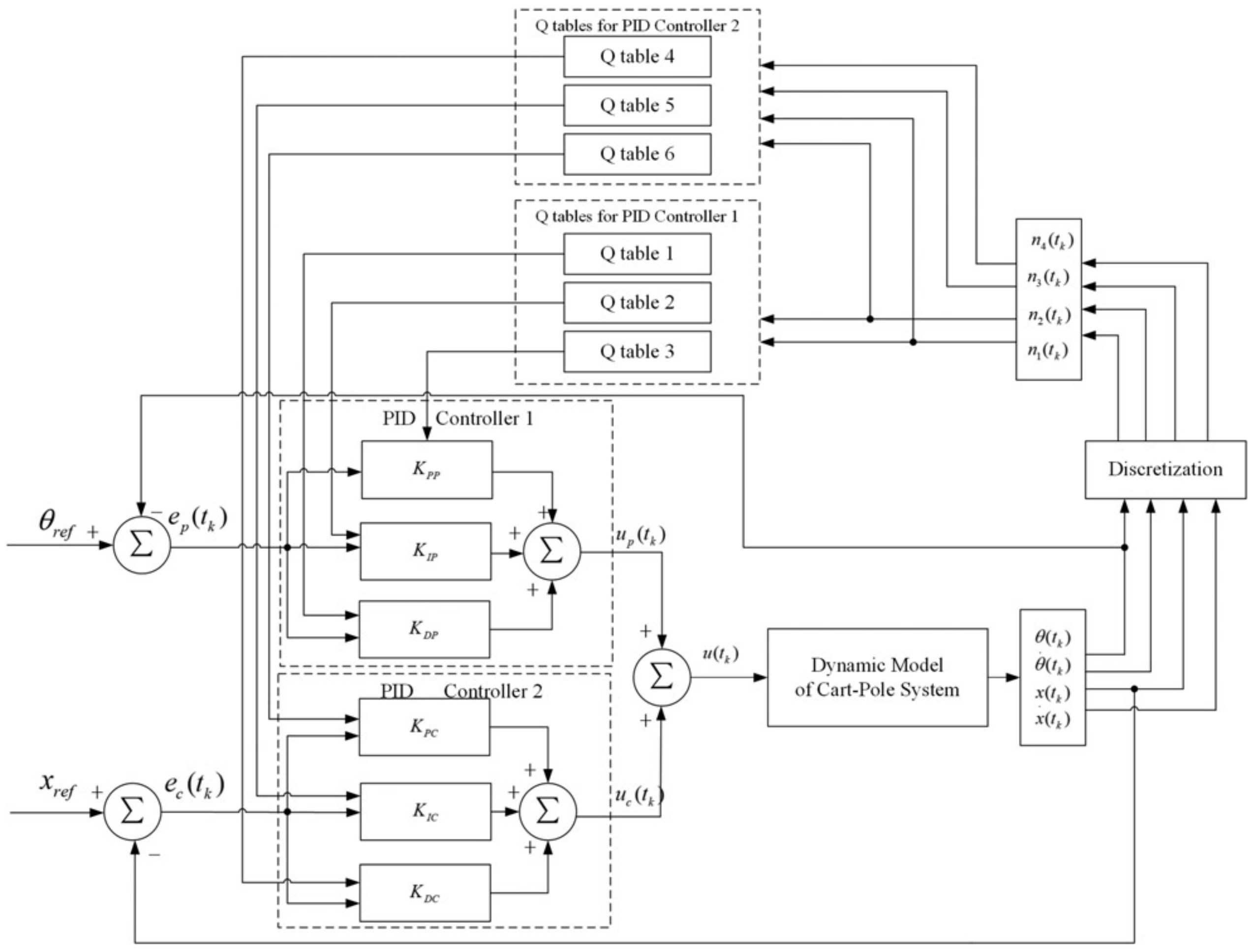

Fig. 5 *Architecture of adaptive PID controller based on Q-learning algorithm*

and outputs the force $u_c(t_k)$, which is defined as

$$u_c(t_k) = K_{\rm PC} e_c(t_k) + K_{\rm IC} \sum_{i=0}^{k} e_c(t_i)\Delta t_k + K_{\rm DC} \frac{e_c(t_k) - e_c(t_{k-1})}{\Delta t_k}, \tag{14}$$

where $K_{\rm PC}$, $K_{\rm IC}$ and $K_{\rm DC}$ are the proportional, integral and derivative gain of PID controller 2, respectively.

The force $u_p(t_k)$ and $u_c(t_k)$ are summed up as a total force $u(t_k)$ which applies to balance the cart–pole system. By simulating the system dynamics, the states of the system $(\theta(t_k), \dot{\theta}(t_k), x(t_k), \dot{x}(t_k))$ are obtained. These four continuous variables are discretised into four discrete variables $(n_1(t_k), n_2(t_k), n_3(t_k), n_4(t_k))$, which are taken as the presentation of state $S_{t_k} \in \mathcal{S}$ in the Q-learning algorithm.

There are six learned Q-tables in total with each Q-table is associated with one controller gain. Q Tables 1–3 are associated with the derivative, integral and proportional gain of the PID controller 1 which controls the pendulum, respectively. Similarly, Q Tables 4–6 are associated with the derivative, integral and proportional gain of the PID controller 2, respectively, which controls the cart, accordingly. To be noted, Q Tables 1–3 only take $n_1(t_k)$ and $n_2(t_k)$ as states, while Q Tables 4–6 take the whole four variables $n_1(t_k)$, $n_2(t_k)$, $n_3(t_k)$ and $n_4(t_k)$ into consideration. When given the current state, each learned Q-table generates the optimal value for the corresponding controller gain.

3.2 Training process

One crucial procedure in designing the controller is the training of the Q-tables, which maps the current states to actions (different values of gains in this situation). The whole training process is shown in pseudo code form Algorithm 2 (see Fig. 6). As to speed up the learning process, an adaptive learning rate method named Delta–Bar–Delta [25] was implemented.

The following are some details of the training process.

3.2.1 Discretisation: Each continuous variable is divided into several buckets, the values within the same bucket are regarded as one same state. The buckets are set using the same rule defined as

$$n = \begin{cases} 1 & \text{if } x_{\rm con} < X_{\rm min} \\ 10 & \text{if } x_{\rm con} > X_{\rm max} \\ \left\lfloor \dfrac{x_{\rm con}}{X_{\rm max} - X_{\rm min}} \times N \right\rfloor + 1 & \text{if } X_{\rm min} \le x_{\rm con} \le X_{\rm max}, \end{cases} \tag{15}$$

where $\lfloor x \rfloor = \max\{n \in \mathbb{Z} | n \le x\}$; n denotes the discrete variable; $x_{\rm con}$ denotes the continuous variable; $X_{\rm min}$ and $X_{\rm max}$ are the lower and upper bounds of $x_{\rm con}$, respectively; and N denotes the number of buckets each variable is divided into, $N = 10$ in this situation. The number of the buckets is decided depending on the simulation performance.

The values set for discretisation are shown in Table 1.

3.2.2 ϵ-greedy method: When given the current state, all six Q-tables generate the actions according to the ϵ-greedy method. This method is defined by

$$A = \begin{cases} \text{random action} & \text{if } \xi < \epsilon \\ \arg\max\limits_{a} Q(s, a) & \text{otherwise} \end{cases}, \tag{16}$$

```
Initialize $Q_i(s,a)=0, \forall a \in \mathcal{A}, \forall s \in \mathcal{S}, i=1,2,...,6$;
Initialize learning rates $\alpha_1$ and $\alpha_2$;
Initialize exploration rate $\epsilon$;
while $episode < maxepisode$ do
    $t=0$;
    Initialize $S_t(\theta(t),\dot{\theta}(t),x(t),\dot{x}(t))$;
    Decay $\epsilon$ (when $episode > 0.6 \times maxepisode$ $\epsilon$=0);
    for $t=1; t \le maxtime; t++$ do
        Discretization of state $S_t$, obtain
          $n_1(t),n_2(t),n_3(t),n_4(t)$;
        for $i=1; i \le 3; i++$ do
            According to $n_1(t),n_2(t)$, choose action $A_i$
              following $\epsilon$-greedy policy;
        end
        for $i=3; i \le 6; i++$ do
            According to $n_1(t),n_2(t),n_3(t),n_4(t)$, choose
              action $A_i$ following $\epsilon$-greedy policy;
        end
        Obtain total output $u(t)$ according to Eq. 11 to 14;
        Observe new state
          $S_{t+1}(\theta(t+1),\dot{\theta}(t+1),x(t+1),\dot{x}(t+1))$;
        Receive reward $R_p$ for
          $Q_1(s,a),Q_2(s,a)\ and\ Q_3(s,a)$;
        Receive reward $R_c$ for
          $Q_4(s,a),Q_5(s,a)\ and\ Q_6(s,a)$;
        Discretization of state $S_{t+1}$, obtain
          $n_1(t+1),n_2(t+1),n_3(t+1),n_4(t+1)$;
        Update learning rate $\alpha_1$ for
          $Q_1(s,a),Q_2(s,a)\ and\ Q_3(s,a)$;
        Update learning rate $\alpha_2$ for
          $Q_4(s,a),Q_5(s,a)\ and\ Q_6(s,a)$;
        Update $Q_1(s,a),Q_2(s,a)\ and\ Q_3(s,a)$ using $R_p$
          and $\alpha_1$;
        Update $Q_4(s,a),Q_5(s,a)\ and\ Q_6(s,a)$ using $R_c$
          and $\alpha_2$;
        $S_t \leftarrow S_{t+1}$
    end
end
```

Fig. 6 *Algorithm 2: the training process*

Table 1 Values set for discretisation

Values	$X_{\min}$	$X_{\max}$
$\theta(t)$	$-45 \times \pi/180$	$45 \times \pi/180$
$\dot{\theta}(t)$	−15	15
$x(t)$	−3	3
$\dot{x}(t)$	−15	15

where ξ is a random number subject to normal distribution, $0 \le \xi \le 1$.

As to speed up the convergence, the value of ϵ decays with the increment of training episodes and is set to zero after certain episodes, while the number of episodes is decided according to the training performance. The details are defined as

$$\epsilon(\text{eps}) = \begin{cases} \dfrac{1}{1+e^{\text{eps}}} + 0.001 & \text{eps} < 0.6 \times \text{maxepisode} \\ 0 & \text{otherwise,} \end{cases} \tag{17}$$

where eps is the current episode and maxepisode is the maximum number of episodes.

3.2.3 Reward scheme: The reward schemes are different according to the PID controllers. The *Q*-tables for pendulum share the same reward scheme, while the others apply a different one. The reward scheme for the *Q*-tables that are associated with the gains of PID controller 1 is defined as

$$R_p = \begin{cases} 1 & \text{if } |\theta(t+1)| < |\theta(t)| \text{ and } |\theta(t+1)| > \theta_{\lim} \\ 2 & \text{if } |\theta(t+1)| \le \theta_{\lim} \\ 0 & \text{otherwise,} \end{cases} \tag{18}$$

where $\theta_{\lim}$ is the threshold set for angle, $\theta_{\lim} = 0.01$ in this situation.

The reward scheme for the *Q*-tables that are associated with the gains of PID controller 2 is defined as

$$R_c = \begin{cases} 1 & \text{if } |\theta(t+1)| < |\theta(t)| \text{ and } |x(t+1)| < |x(t)| \text{ and } |x(t+1)| > x_{\lim} \\ 2 & \text{if } |x(t+1)| \le x_{\lim} \\ 0 & \text{otherwise,} \end{cases} \tag{19}$$

where $x_{\lim}$ is the threshold set for position, $x_{\lim} = 0.1$ in this situation.

3.2.4 Adaptive learning rate: As to improve the efficiency of convergence, the adaptive learning rate algorithm called Delta–Bar–Delta [25] was implemented. The algorithm is defined as

$$\Delta\alpha_t = \begin{cases} \kappa & \text{if } \bar{\delta}_{t-1}\delta_t > 0 \\ -\phi\alpha_t & \text{if } \bar{\delta}_{t-1}\delta_t < 0 \\ 0 & \text{if } \bar{\delta}_{t-1}\delta_t = 0 \end{cases} \tag{20}$$

where $\Delta\alpha_t$ is the increment of the learning rate in time step *t*; κ is a positive constant value to increase the learning rate; ϕ is a positive constant value denoting the discounting factor; δ_t is the temporal difference (TD) error [22] in time step *t*, $\delta_t = R_{t+1} + \gamma \max Q(S_{t+1}, a) - Q(S_t, A_t)$; $\bar{\delta}_t = (1-\varphi)\delta_t + \varphi\bar{\delta}_{t-1}$.

The learning rate in time step $t+1$ is updated as

$$\alpha_{t+1} = \alpha_t + \Delta\alpha_t. \tag{21}$$

By applying this rule, the learning rate will be updated by comparing the current TD error with the accumulated TD error in previous steps. When the learning rate becomes too large, the increment of the learning rate changes sign and reduces the learning rate. On the other hand, if the learning rate is too small, the learning rate keeps changing in the previous trend and speeds up the convergence.

All *Q*-tables use adaptive learning rate algorithm, while the setting of parameters is different. *Q*-tables used to change the gains of the controller of the pendulum share the same set of parameters, while the other *Q*-tables share another set of parameters. The values used in this research are presented in Table 2.

3.2.5 Other settings of parameters: The parameters used in initialisation are listed in Table 3. The running of each episode will be interrupted if one of the following situations occur, $|\theta(t)| > 0.5\pi$ or $|x(t)| > 10.0$, which is not ideal in real physical world as to improve the efficiency of the simulation.

4 Simulation results

In this section, the simulation results of implementing adaptive PID controller based on *Q*-learning algorithm (named as QPID controller

Table 2 Parameters set for adaptive learning rate algorithm

	Q tables of PID controller 1	Q tables of PID controller 2
α_0	0.015	0.3
κ	$0.1\alpha_0$	$0.002\alpha_0$
ϕ	0.5	0.5
φ	0.5	0.5

Table 3 Parameters set in initialisation

Variables	Values
initial position of $\theta(t)$	$-0.7854, +0.7854$
initial position of $\dot{\theta}(t)$	0
initial position of $x(t)$	$-0.1, +0.1$
initial position of $\dot{x}(t)$	0
number of actions for all tables	50
variation range of K_{PP}	[–3000, 0]
variation range of K_{IP}	[–300, 0]
variation range of K_{DP}	[–3000, 0]
variation range of K_{PC}	[–2000, 0]
variation range of K_{IC}	[–5, 0]
variation range of K_{DC}	[0, 2000]
maximum episodes	6000
maximum timesteps	8000
interval sampling time	10 ms
γ	0.99

Table 4 Parameters of cart–pole system

Variables	Values
M	1.3282 kg
m	0.22 kg
l	0.304 m
F_0	22.915 N/m/s
F_1	0.007056 N/rad/s
g	9.8 m/s^2

Table 5 Parameters of Q controller

Variables	Values
initial position of $\theta(t)$	0.1745
initial position of $\dot{\theta}(t)$	0
initial position of $x(t)$	0
initial position of $\dot{x}(t)$	0
maximum episodes	30,000
maximum time step	2000
interval sampling time	10 ms
discount factor γ	0.99
learning rate α	0.01
exploration rate ϵ	min (1500/episode)

Table 6 Parameters of PID controllers

Gains	PID controller 1	PID controller 2
P	–270	–80
I	–10	–0.5
D	–1500	2000

in the following content for the simplicity of expression) are demonstrated. The QPID controller was trained from four fixed initial positions $(\theta(t) = \pm 0.7854\,\text{rad},\ \dot{\theta}(t) = 0,\ x(t) = \pm 0.05\,\text{m},\ \dot{x}(t) = 0)$ with 0–0.04 rad subtracted from each fixed point randomly on angle as disturbances. The learned Q-tables were implemented to generate the optimal gains of the PID controllers. The QPID controller was applied to balance the cart–pole system from a series of initial positions, which are the combinations of

$$\theta(t) \in \{\pm 0.7854,\ \pm 0.5236,\ \pm 0.2618,\ 0\,\text{rad}\},$$

$$\dot{\theta}(t) = 0,\ x(t) \in \{\pm 0.05,\ \pm 0.02,\ 0\,\text{m}\},\ \dot{x}(t) = 0$$

and succeeded in balancing the system from all these initial positions. Additionally, as for comparison, two other types of controllers are applied to the system to compare the controlling performances. The first type of controller employs the tabular Q-learning algorithm (named as Q controller in the following content for the simplicity of expression) and the second one is the conventional PID controller. Position $\theta(t) = 0.1745\,\text{rad}$, $\dot{\theta}(t) = 0,\ x(t) = 0,\ \dot{x}(t) = 0$ was chosen as the initial position to compare the performance of QPID controller with Q controller, considering that the Q controller has the limitation and is not able to balance the system when the angle or the position exceeds the set value. Position $x_1 = 45°,\ x_2 = 0,\ x_3 = 0.05\,\text{m},\ x_4 = 0$ was chosen as the initial position to test the performance of QPID controller against the PID controller. The structure of the following content is as follows. Section 4.1 provides the setting of the parameters in the simulation, which includes parameters set for the cart–pole system and the controllers used in comparison. Section 4.2 compares the performance of QPID controller against Q controller and the performance of the QPID controller against PID controller, separately.

4.1 Setting of parameters

4.1.1 Cart–pole system: The parameters set for the cart–pole system are illustrated in Table 4.

4.1.2 Q controller: The Q controller takes the four variables of the cart–pole system $(\theta(t), \dot{\theta}(t), x(t), \dot{x}(t))$ as input and generates the force which directly applies to the cart to balance the system. $\theta(t)$ is divided into six buckets (with $-21 \times \pi/180$, $-6 \times \pi/180,\ 0,\ 6 \times \pi/180,\ 21 \times \pi/180$ as dividing points); $\dot{\theta}(t)$ is divided into three buckets (with $\pm 6 \times \pi/180$ as dividing points); $x(t)$ is divided into three buckets (with ± 2.4 as dividing points); $\dot{x}(t)$ is divided into three buckets (with ± 0.5 as dividing points). The output force is mapped into ten actions which evenly distribute among $[-100N,\ +100N]$. The system

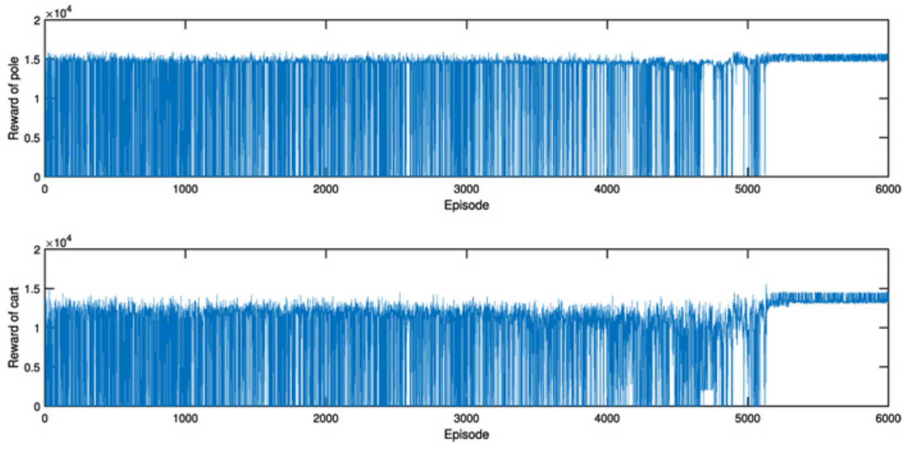

Fig. 7 *Learning curve of QPID controller*

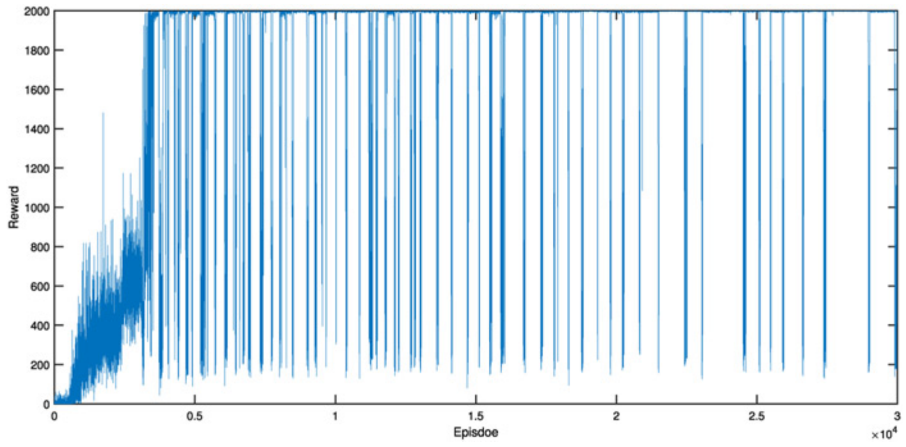

Fig. 8 *Learning curve of Q Controller*

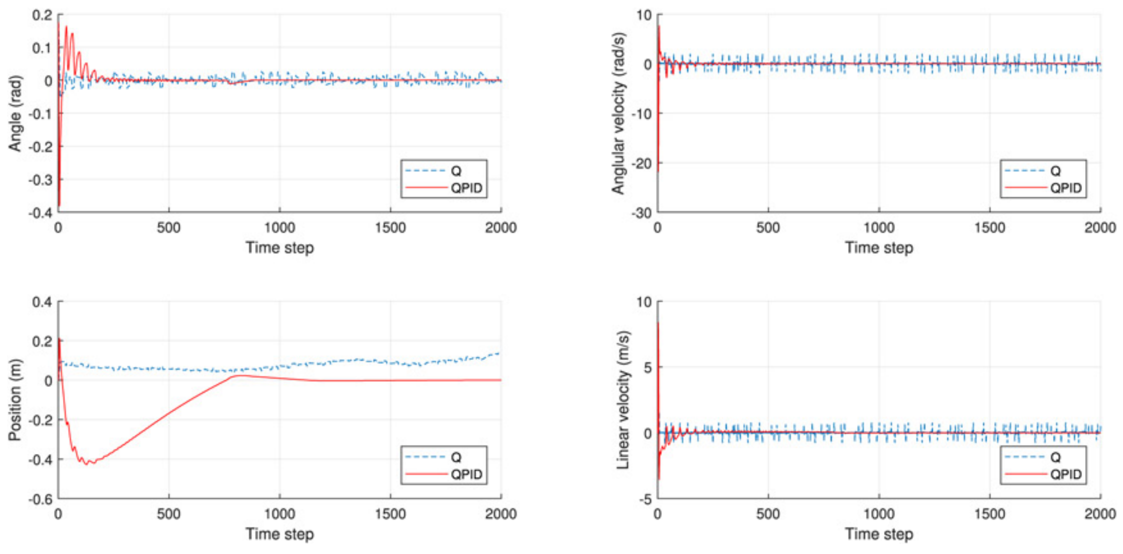

Fig. 9 *Response curves of Q controller and QPID controller*

receives reward as 1 when $|\theta(t)| < 6 \times \pi/180$ or $|x(t)| < 0.1$, otherwise, reward is 0.

The relevant parameters set for Q controller is given in Table 5.

4.1.3 PID controller: The PID scheme applied in this research consists of two linear PID controllers, which control the angle position of the pole and the position of the cart separately. The gains of two linear PID controllers were obtained by manual tuning according to Ziegler–Nichols method [9] and are shown in Table 6.

4.2 Controlling performance

4.2.1 Learning curves: The learning performance of Q-learning algorithm is indicated by the learning curve, which plots the accumulated reward obtained with the incrementation of the episode. The aim of comparing learning curves is to indicate the learning efficiency of Q-learning algorithm applied in different controllers. The convergence of the learning curve shows the success in Q-learning algorithm and the time steps used to reach convergence indicates the learning efficiency. The learning curve of QPID controller in the training process is presented in Fig. 7 and the learning curve of the Q controller is shown in Fig. 8, both of which show the convergence. However, it can be observed that the learning curve of the QPID controller does not show the convergence trend as the conventional learning curves do. This is due to the actions chose by the Q-tables are the gains of PID controller instead of the forces. There are several combinations of gains which would be suitable for the PID controller to balance the system and reduce the difficulty of the Q-learning algorithm in finding proper set of gains for the PID controllers, which results in the high frequency of appearance of high accumulated reward in the learning curve. Nevertheless, after about 5000 time steps, the exploration rate of the Q-learning algorithm used in QPID controller was set to zero as to improve the efficiency of converge; an obvious convergence can be noticed, which indicates the success of Q-learning algorithm in finding optimal solutions to adapting the gains.

Table 7 Characteristics of response curves (Q controller and QPID controller)

Characteristics	*Q* controller	QPID controller
overshoot (angle)	0.0479 (rad)	0.3763 (rad)
undershoot (angle)	0.1316 (rad)	0.1574 (rad)
rise time (angle)	6 (time steps)	4 (time step)
settling time (angle)	4 (time steps)	69 (time step)
steady error(angle)	0.026 (rad)	0 (rad)
overshoot (position)	0.0930 (m)	0.2121 (m)
undershoot (position)	0.2443 (m)	0.4264 (m)
rise time (position)	76 (time steps)	86 (time step)
settling time (position)	—	772 (time step)
steady error (position)	0.1351 (m)	0 (m)

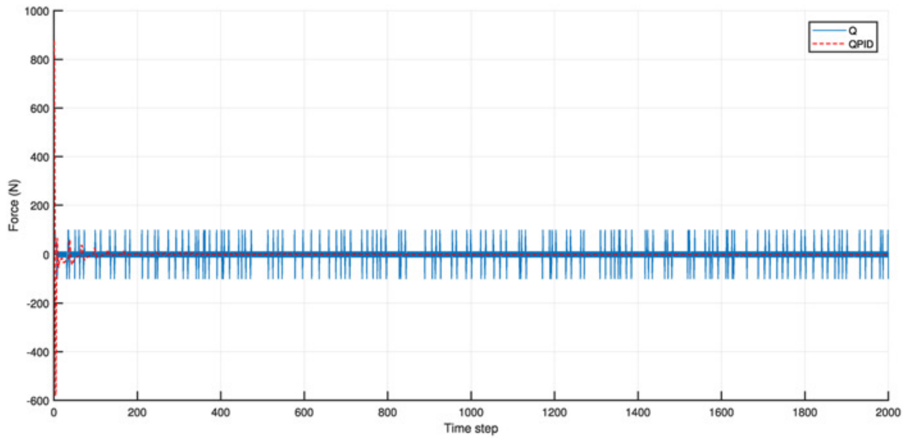

Fig. 10 *Forces generated by Q controller and QPID controller*

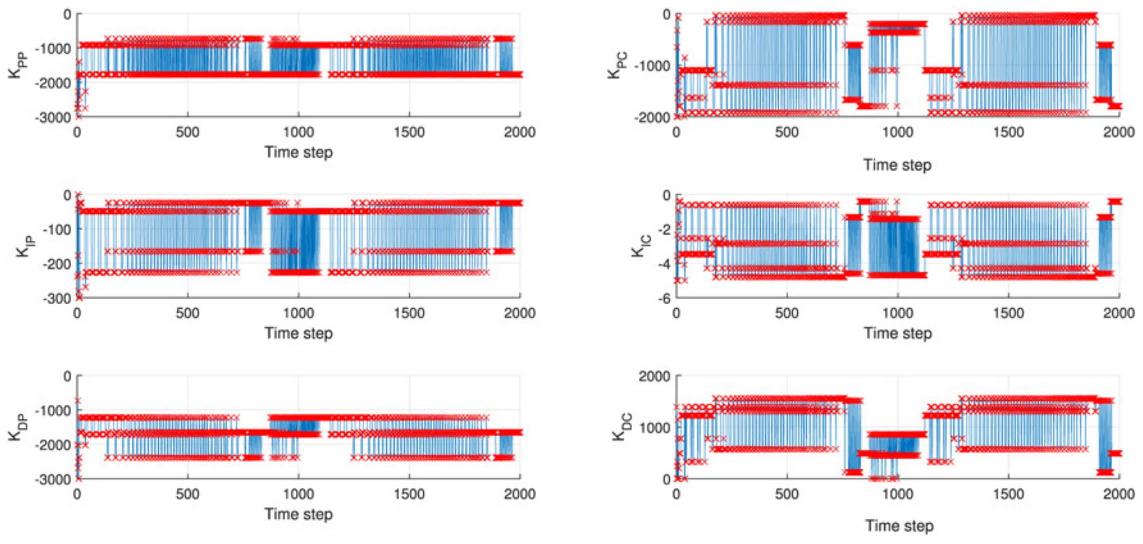

Fig. 11 *Adaption of the gains of QPID controllers (initial position $\theta(t) = 0.1745$ rad, $\dot{\theta}(t) = 0$, $x(t) = 0$, $\dot{x}(t) = 0$)*

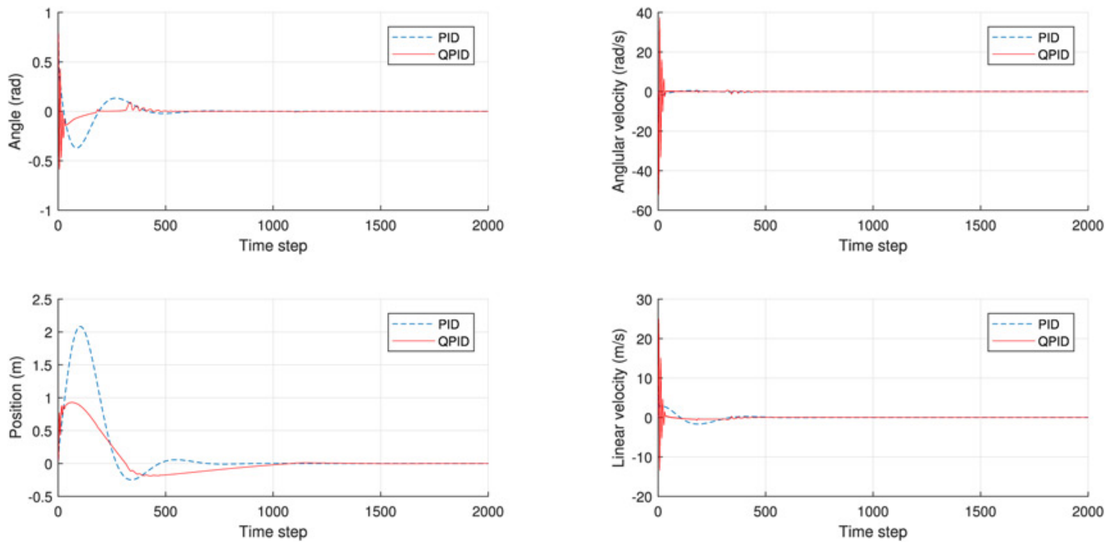

Fig. 12 *Response curves of PID controller and QPID controller*

4.2.2 Performances of controllers: The learned Q-tables were implemented in the designed controllers to balance the system. The following shows the comparison of different controlling performances by plotting the response curves of cart–pole system applied with different controllers. In each comparison, four response curves are given by plotting the position and velocity of both the pole and the cart against time in one figure, which indicates the characteristics of the controllers.

(i) Comparison of Q controller with QPID controller: Fig. 9 compares the response curve of the system controlled by Q controller and QPID controller with the initial position $\theta(t) = 0.1745\,\text{rad}$, $\dot{\theta}(t) = 0$, $x(t) = 0$, $\dot{x}(t) = 0$. The characteristics of the response curves are shown in Table 7. The overshooting or undershooting of the curves related to the QPID controller are greater than the ones related to the Q controller both in pole and in cart positions. Besides, the settling time of the QPID controller (69 time steps for pole and 86 time steps for cart) is longer than the Q controller (4 time steps for pole and 76 time steps for cart). One factor that leads to these performances is the initial position in this test was not trained in the learning process, the combination of the gains might not be the optimal solution to the controllers from this starting position. However, QPID shows an obvious advantage in steady state since Q PID controller reaches the desired position both for the pole and for the cart, while Q controller remains an error of 0.026 rad for pole and 0.1351 m for cart with obvious oscillation. In conclusion, the Q PID controller has bigger overshooting and undershooting and longer settling time compared with Q controller, but it has better performance in reaching steady state and reducing steady-state error.

The total forces generated by two controllers are shown in Fig. 10. The adaption of the gains of QPID controller during controlling process is presented in Fig. 11, where the red crosses denote the chosen values of the gains of PID controllers in each time step t and the blue lines indicate the trend of variation.

(ii) Comparison of PID controller with QPID controller: The initial position set in this comparison is $\theta(t) = 0.7854\,\text{rad}$, $\dot{\theta}(t) = 0$, $x(t) = 0.1\,\text{m}$, $\dot{x}(t) = 0$. Fig. 12 shows the response curves of the cart–pole system when QPID controller and PID controller were applied to control. The characteristics of the response curves are illustrated in Table 8. The QPID controller results in bigger overshoot in balancing the pole while smaller overshoot and undershoot in balancing the position of the cart. The settling time in both QPID controller and PID controller is almost the same.

Table 8 Characteristics of response curves (PID controller and QPID controller)

Characteristics	PID	QPID
overshoot (angle)	0.3527 (rad)	0.5866 (rad)
undershoot (angle)	0.1316 (rad)	0.4306 (rad)
rise time (angle)	44 (time steps)	4 (time step)
settling time (angle)	320 (time steps)	64 (time step)
steady error (angle)	0 (rad)	0 (rad)
overshoot (position)	2.075 (m)	0.9279 (m)
undershoot (position)	0.2443 (m)	0.1807 (m)
rise time (position)	76 (time steps)	13 (time step)
settling time (position)	426 (time steps)	772 (time step)
Steady error (position)	0 (m)	0 (m)

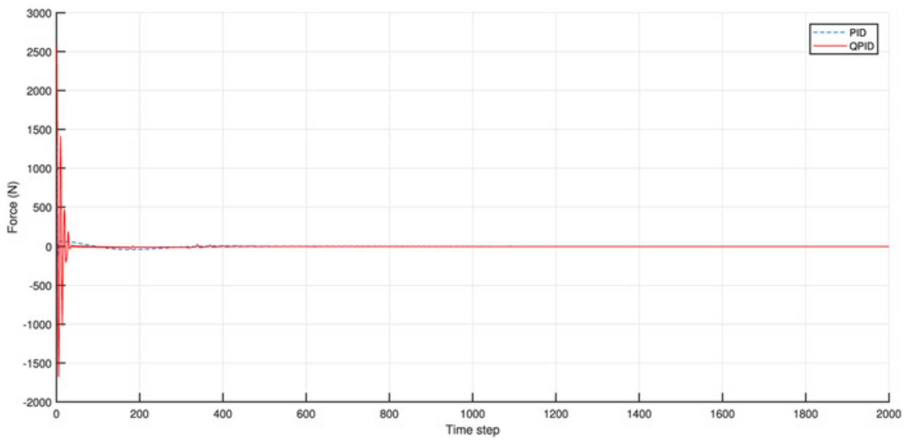

Fig. 13 *Forces generated by PID controller and QPID controller*

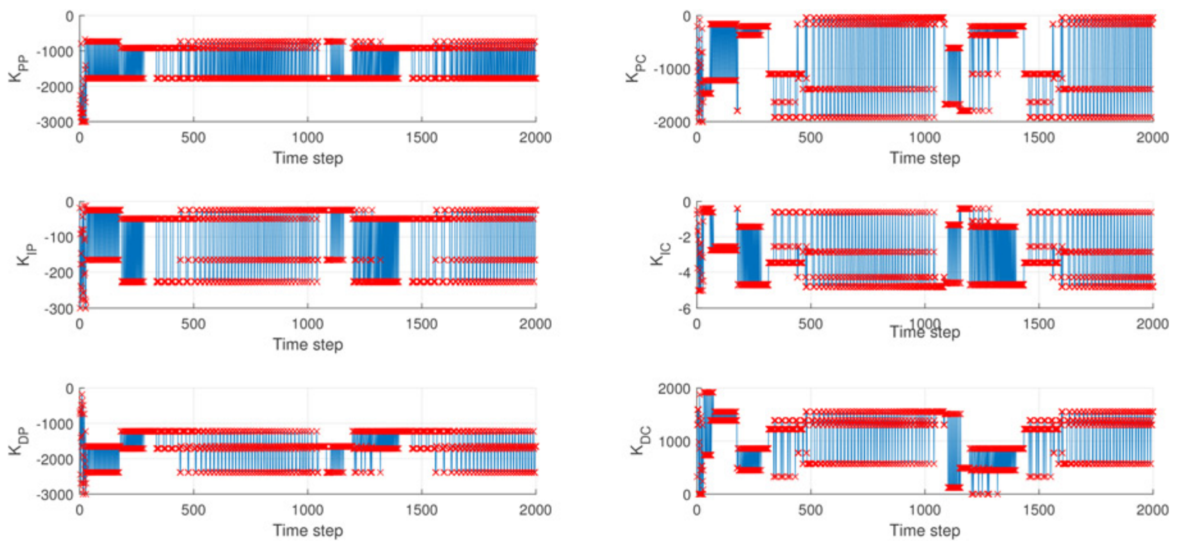

Fig. 14 *Adaption of the gains of QPID controllers (initial position* $\theta(t) = 0.7854\,\text{rad}$, $\dot{\theta}(t) = 0$, $x(t) = 0.1\,\text{m}$, $\dot{x}(t) = 0$*)*

The output forces generated by QPID controller and PID controller are shown in Fig. 13.

Fig. 14 demonstrates the adaption of the gains of QPID controller during controlling process.

5 Conclusion

In this research, an adaptive PID controller based on *Q*-learning algorithm was proposed. The performance of the controller was tested on the cart–pole problem with comparison of conventional PID controller and *Q* controller. According to the simulation results, QPID controller designed in this research was trained in four fixed positions and was able to balance the system starting from various different initial positions, which shows the advantage in generality of adapting the changes made to systems. Two fixed initial positions were chosen to compare controlling performance with the conventional PID controller and the *Q* controller, where the QPID controller shows relatively equivalent performance or even better in stabilisation of achieving smaller steady-state error. However, there are tiny oscillations observed and obvious overshooting and undershooting in the controlling performance using QPID controller, which could lead to unstable behaviour of the system and need to be improved in the future research. Besides, the learning efficiency is considered to be improved. Additionally, future research plans to apply this QPID controller to various different applications, such as the mass–spring–damper system, beam and ball system, multiple-link arms, etc., to improve the generality.

6 Acknowledgments

This work was partly supported by King's College London, China Scholarship Council and the Ministry of Science and Technology, Taiwan (grant MOST-106-2221-E-027-009).

7 References

[1] Miranda, M.F., Vamvoudakis, K.G.: 'Online optimal auto-tuning of PID controllers for tracking in a special class of linear systems'. American Control Conf. (ACC), Boston, MA, 2016, pp. 5443–5448

[2] Carlucho, I., De Paula, M., Villar, S.A., *et al.*: 'Incremental Q-learning strategy for adaptive PID control of mobile robots', *Expert Syst. Appl.*, 2017, **80**, pp. 183–199

[3] Shadi, M., Sargolzaei, M.: 'Application of reinforcement learning to improve control performance of plant'. IEEE Int. Conf. Computational Intelligence for Measurement Systems and Applications, Istanbul, 2008

[4] Lillicrap, T.P., Hunt, J.J., Pritzel, A., *et al.*: 'Continuous control with deep reinforcement learning', arXiv preprint arXiv:1509.02971, 2015

[5] Bagnell, J.A., Schneider, J.G.: 'Autonomous helicopter control using reinforcement learning policy search methods'. Proc. IEEE Int. Conf. on Robotics and Automation, Seoul, South Korea, 2001, vol. 2, pp. 1615–1620

[6] Fernandez-Gauna, B., Ansoategui, I., Etxeberria-Agiriano, I., *et al.*: 'Reinforcement learning of ball screw feed drive controllers', *Eng. Appl. Artif. Intell.*, 2014, **30**, pp. 107–117

[7] Ramanathan, P., Mangla, K.K., Satpathy, S.: 'Smart controller for conical tank system using reinforcement learning algorithm', *Measurement*, 2018, **116**, pp. 422–428

[8] Boubertakh, H., Tadjine, M., Glorennec, P.-Y., *et al.*: 'Tuning fuzzy pd and pi controllers using reinforcement learning', *ISA Trans.*, 2010, **49**, (4), pp. 543–551

[9] Åström, K.J., Hägglund, T.: 'PID controllers: theory, design, and tuning', vol. 2 (Instrument Society of America, Research Triangle Park, NC, 1995)

[10] Wu, C.J., Lee, T.L., Fu, Y.Y., *et al.*: 'Auto-tuning fuzzy PID control of a pendubot system'. 2007 IEEE Int. Conf. on Mechatronics, Changchun, Jilin, May 2007, pp. 1–6

[11] Duan, H.B., Wang, D.B., Yu, X.F.: 'Novel approach to nonlinear pid parameter optimization using ant colony optimization algorithm', *J. Bionic Eng.*, 2006, **3**, (2), pp. 73–78

[12] Varol, H.A., Bingul, Z.: 'A new PID tuning technique using ant algorithm'. Proc. 2004 American Control Conf., Boston, MA, USA, June 2004, vol. 3, pp. 2154–2159

[13] Howell, M.N., Frost, G.P., Gordon, T.J., *et al.*: 'Continuous action reinforcement learning applied to vehicle suspension control', *Mechatronics. (Oxf)*, 1997, **7**, (3), pp. 263–276

[14] Mohammadi, S.M.A., Gharaveisi, A.A., Mashinchi, M., *et al.*: 'New evolutionary methods for optimal design of PID controllers for AVR system'. 2009 IEEE Bucharest PowerTech, Bucharest, June 2009, pp. 1–8

[15] Pour, F.M., Gharaveisi, A.A.: 'Opposition-based discrete action reinforcement learning automata algorithm case study: optimal design of a PID controller', *Turk. J. Electr. Eng. Comput. Sci.*, 2013, **21**, (6), pp. 1603–1614

[16] Howell, M.N., Best, M.C.: 'On-line PID tuning for engine idle-speed control using continuous action reinforcement learning automata', *Control Eng. Pract.*, 2000, **8**, (2), pp. 147–154

[17] Mohammadi, S., Gharaveisi, A., Mashinchi, M., *et al.*: 'Development of a novel reinforcement learning automata method for optimum design of proportional integral derivative controller for nonlinear systems'. Proc. World Congress on Engineering, London, UK, 2008

[18] Sedighizadeh, M., Rezazadeh, A.: 'Adaptive PID controller based on reinforcement learning for wind turbine control'. *International Scholarly and Scientific Research & Innovation*, 2008, **2**, (1), pp. 124–129

[19] Watkins, C.J., Dayan, P.: 'Q-learning', *Mach. Learn.*, 1992, **8**, (3–4), pp. 279–292

[20] Esmaeili, M., Shayeghi, H., Mohammad Nejad, H., *et al.*: 'Reinforcement learning based PID controller design for LFC in a microgrid', *COMPEL-Int. J. Comput. Math. Electr. Electron. Eng.*, 2017, **36**, (4), pp. 1287–1297

[21] el Hakim, A., Hindersah, H., Rijanto, E.: 'Application of reinforcement learning on self-tuning PID controller for soccer robot multi-agent system'. 2013 Joint Int. Conf. on Rural Information Communication Technology and Electric-Vehicle Technology (rICT ICeV-T), Bandung, November 2013, pp. 1–6

[22] Sutton, R.S., Barto, A.G.: 'Reinforcement learning: an introduction', vol. 1 (MIT Press, Cambridge, 1998)

[23] Watkins, C.J.C.H.: 'Learning from delayed rewards'. PhD thesis, King's College, Cambridge, 1989

[24] Lam, H.K., Leung, H.F.H.: 'Fuzzy controller with stability and performance rules for nonlinear systems', *Fuzzy Sets Syst.*, 2007, **158**, (2), pp. 147–163

[25] Jacobs, R.A.: 'Increased rates of convergence through learning rate adaptation', *Neural Netw.*, 1988, **1**, (4), pp. 295–307