mathematics

MDPI

*Article*

# Adaptive Proportional Integral Robust Control of an Uncertain Robotic Manipulator Based on Deep Deterministic Policy Gradient

**Puwei Lu, Wenkai Huang *, Junlong Xiao, Fobao Zhou and Wei Hu**

School of Mechanical & Electrical Engineering, Guangzhou University, Guangzhou 510006, China; 2111907042@e.gzhu.edu.cn (P.L.); 1707200071@e.gzhu.edu.cn (J.X.); 1807700048@e.gzhu.edu.cn (F.Z.); 1707700083@e.gzhu.edu.cn (W.H.)

* Correspondence: smallkat@gzhu.edu.cn

**Abstract:** An adaptive proportional integral robust (PIR) control method based on deep deterministic policy gradient (DDPGPIR) is proposed for n-link robotic manipulator systems with model uncertainty and time-varying external disturbances. In this paper, the uncertainty of the nonlinear dynamic model, time-varying external disturbance, and friction resistance of the n-link robotic manipulator are integrated into the uncertainty of the system, and the adaptive robust term is used to compensate for the uncertainty of the system. In addition, dynamic information of the n-link robotic manipulator is used as the input of the DDPG agent to search for the optimal parameters of the proportional integral robust controller in continuous action space. To ensure the DDPG agent's stable and efficient learning, a reward function combining a Gaussian function and the Euclidean distance is designed. Finally, taking a two-link robot as an example, the simulation experiments of DDPGPIR and other control methods are compared. The results show that DDPGPIR has better adaptive ability, robustness, and higher trajectory tracking accuracy.

**Keywords:** n-link robot; deep deterministic policy gradient; adaptive control; proportional integral robust control; reward function



**Citation:** Lu, P.; Huang, W.; Xiao, J.; Zhou, F.; Hu, W. Adaptive Proportional Integral Robust Control of an Uncertain Robotic Manipulator Based on Deep Deterministic Policy Gradient. *Mathematics* **2021**, *9*, 2055. https://doi.org/10.3390/math9172055

Academic Editor: António M. Lopes

Received: 18 July 2021
Accepted: 25 August 2021
Published: 26 August 2021



## 1. Introduction

A robotic manipulator is similar to the human arm and can replace or assist humans to complete the tasks of picking, placing, painting, welding, and assembling. The manipulator plays an important role in industrial production, underwater exploration, medical application, aerospace, and other fields [1–4]. To achieve a better control effect and meet the control requirements of different fields, the manipulator must have the ability to track a trajectory with high precision. Due to the highly nonlinear, dynamic characteristics of a robotic manipulator, and the influence of joint friction and time-varying external interference in practical applications, it is difficult to obtain accurate information about model parameters. Therefore, when designing a control strategy, good adaptability and high-precision trajectory tracking abilities are necessary for the uncertainty of the n-link robotic manipulator system.

In order to better control the robot manipulator, the robustness of the control strategy has attracted extensive attention. Robustness here refers to the ability to produce good dynamic behavior in the face of modelling errors and unmodelled dynamics of the robot manipulator [5,6]. Loucif and Kechida [7] and Elkhateeb et al. [8] used a whale optimization algorithm and an artificial bee colony algorithm, respectively, to optimize the parameters of the proportion integral differential (PID) controller, improve the trajectory tracking accuracy of the robot manipulator under unmodeled dynamics, and make the controller have a certain robustness. In order to model the control process of the robot manipulator more accurately, Ardeshiri et al. [9,10] proposed a fractional order fuzzy PID controller.

A fractional order controller summarizes the design of an integer order PID controller and extends it from point to plane. This extension increases the flexibility of control system design and can realize the control process more accurately. With the help of a fractional order PID controller, the controller can be designed to ensure that the closed-loop system has a stronger robustness to gain variation and aging effect [11]. Therefore, PID and other model-based control strategies have proved to be effective, but these methods need to obtain dynamic model information of the controlled object. In the actual control process, it is difficult to obtain accurate information due to the complexity of the manipulator mechanism and the uncertainty of external interference [12,13]. To solve this problem, it is necessary to compensate or approximate the uncertainty and time-varying external disturbance of the nonlinear dynamic model to meet the demand of actual control. Wang [14] used the robust controller to compensate for the uncertainty, unmodeled dynamics, and external interference of the dynamic model parameters of the robot manipulator, and so realized accurate tracking for it. Yang and Jiang [15] and Rouhani [16] used a fuzzy logic system to approximate the nonlinear dynamic model of the robot manipulator. However, the design of the fuzzy logic system depends on expert knowledge. A neural network is good at approximating the uncertain mathematical model, and it is one of the effective solutions to nonlinear system control problems. Yang et al. [17] proposed an adaptive neural network control method based on a nonlinear observer. The joint speed of the manipulator is estimated by a nonlinear observer, and, based on the estimated value of the speed, an adaptive radial basis function neural network is used to compensate for the uncertainty of the robotic manipulator system, to improve the tracking accuracy of its end force and its joint position. Guo et al. [18] proposed an adaptive neural network control method, which uses the weighted performance function to control the joint angle and the trajectory tracking error within an expected range, approximates the dynamic model of the manipulator through the radial basis function neural network, and uses the adaptive law to adjust the weights of the neural network, to improve the robustness of the controller. Although the neural network has a good compensation or approximation effect for the uncertainty and time-varying disturbance of the nonlinear dynamic model, training the network is likely to converge to the local optimal problem. Therefore, a robust term based on deep reinforcement learning is proposed to compensate for the modeling error of the nonlinear dynamic model of an n-link robot manipulator. Under conditions of structural parameter perturbation, time-varying external interference, and friction resistance, the influence of the uncertainty of the dynamic model on the controller can be reduced so as to maintain the stability of the control system and improve the trajectory tracking performance of an n-link robotic manipulator.

As an important branch of artificial intelligence technology, reinforcement learning mainly selects actions through interactive learning between agents and the environment. The environment responds to the actions of agents and transforms them into a new state. At the same time, it generates a reward. The agent's goal is to maximize the accumulated discount reward value [19,20]. Compared with the classical control method, reinforcement learning does not need to obtain an accurate dynamic model, which is very advantageous in solving the decision sequence problem under highly nonlinear and uncertain conditions [21]. Kukker and Sharma [22] and Runa et al. [23] used the fuzzy Q-learning algorithm to realize trajectory tracking control of the manipulator. Kim et al. [24] used the State-Action-Reward-State-Action (SARSA) algorithm to locate fixed and random target points of the end effector of a three-link Planar Arm. Although Q-learning and SARSA can effectively solve some typical reinforcement learning tasks, the algorithms need to be built in discrete space. Therefore, in the control problem, it is often necessary to discretize the continuous process. However, sparse discretization can easily reduce the control accuracy, and dense discretization can easily fall into the curse of the dimension problem [25,26]. One method for solving this problem is to effectively combine deep learning with reinforcement learning. A deep neural network is used, in traditional reinforcement learning, to model solutions to continuous reinforcement learning tasks [27,28]. Based on this method,

Lillicrap et al. [29] proposed a depth deterministic strategy gradient algorithm based on the actor critic framework. Shi et al. [30] used the DDPG algorithm to deal with controlling the zinc electro winning (Zep) process, which effectively solved the problems of inaccurate modeling and time delay, while also being more energy-saving than the traditional control method. Sun et al. [31] used this algorithm to solve the heavy vehicle adaptive cruise decision-making problem, which has good adaptability in a strange and complex environment. Zhao et al. [32] solved the cooperative control problem of wind farms through the DDPG algorithm and reduced the learning cost in the learning process. Therefore, the DDPG algorithm seems to be effective in solving multiple continuous-state space reinforcement learning tasks.

The purpose of this paper is to establish an n-link robotic manipulator control system with a model for uncertainty and time-varying external disturbances. An adaptive PIR control method based on deep reinforcement learning is proposed. The modeling error of the nonlinear dynamic model of an n-link manipulator is compensated by robust control, and the parameters of the controller are adjusted by a DDPG algorithm to improve the adaptability of the controller to the uncertain, nonlinear, dynamic model. The main contributions of this paper are as follows:

- Considering the uncertainty and time-varying disturbance of the dynamic model of the n-link robot manipulator system and the influence of friction resistance, the adaptive robust term is used to compensate for the uncertainty of the system. An adaptive PIR control method based on the DDPG is proposed, which has good adaptability and high-precision trajectory tracking ability for the uncertainty of the n-link robot manipulator system.
- A reward function combining a Gaussian function and the Euclidean distance is proposed, which can ensure the reinforcement learning agent learns efficiently and stably and can effectively avoid a convergence of the deep neural network to the local optimal problem.
- Taking a two-link robotic manipulator as an example, the simulation results show that the proposed method is effective compared with an adaptive control based on radial basis function neural network (RBFNN) approximation and PIR control with fixed parameters.

## 2. Dynamic Model of the n-Link Robot Manipulator

The dynamic model of the n-link robotic manipulator system expresses the relationship between the joint torque and the position, velocity, and acceleration of the connecting rod:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + F_f(\dot{q}) + \tau_d = \tau \tag{1}$$

where $q \in \mathbb{R}^n$ is the joint position vector of the manipulator, $\dot{q} \in \mathbb{R}^n$ is the velocity vector of the manipulator, $\ddot{q} \in \mathbb{R}^n$ is the acceleration vector of the manipulator, $M(q) \in \mathbb{R}^{n\times n}$ is the mass inertia matrix, $C(q,\dot{q}) \in \mathbb{R}^{n\times n}$ is the Coriolis force and the centrifugal force vector, $G(q) \in \mathbb{R}^{n\times n}$ is the gravity vector, $F_f(\dot{q}) \in \mathbb{R}^n$ is the friction vector, $\tau_d \in \mathbb{R}^n$ is the time-varying external disturbance, and $\tau \in \mathbb{R}^n$ is the torque vector acting on the joint.

The precise values of the $M(q)$, $C(q,\dot{q})$, and $G(q)$ parameters in the dynamic model are difficult to obtain due to a series of influential factors, such as the complexity of the manipulator mechanism, environmental variations, and measurement errors in the actual operation of the manipulator. Therefore, the actual values for $M(q)$, $C(q,\dot{q})$, and $G(q)$ are divided into the model part and the error part as follows:

$$M(q) = M_0(q) + \Delta E_M(q) \tag{2}$$

$$C(q,\dot{q}) = C_0(q,\dot{q}) + \Delta E_C(q,\dot{q}) \tag{3}$$

$$G(q) = G_0(q) + \Delta E_G(q) \tag{4}$$

The dynamic model Formula (1) of the n-link robot manipulator can also be expressed as follows:

$$M_0(q)\ddot{q} + C_0(q,\dot{q})\dot{q} + G_0(q) + E(q,\dot{q}) + F_f(\dot{q}) + \tau_d = \tau \tag{5}$$

$$E(q,\dot{q}) = \Delta E_M(q)\ddot{q} + \Delta E_C(q,\dot{q})\dot{q} + \Delta E_G(q) \tag{6}$$

**Property 1** ([33]). *The mass inertia matrix $M_0(q)$ is symmetric, positive definite and bounded and can be expressed as follows:*

$$\mu_m < \|M_0(q)\| < \mu_n \tag{7}$$

*where $\|M_0(q)\|$ is the norm of the mass inertia matrix $M_0(q)$; $\mu_n$ and $\mu_m$ are the upper and lower boundaries, respectively, and both are positive numbers.*

**Property 2** ([34]). *Coriolis force and centrifugal force matrix is $C_0(q,\dot{q})$. The following equation is satisfied:*

$$\xi^T\left(\dot{M}_0(q) - 2C_0(q,\dot{q})\right)\xi = 0 \tag{8}$$

*Among them, $\left(\dot{M}_0(q) - 2C_0(q,\dot{q})\right)$. It is a skew symmetric matrix, $\xi \in \mathbb{R}^n$.*

**Property 3** ([35]). *The gravity vector $G(q)$ satisfies $\|G(q)\| < \rho$, $\rho \in (0,\infty)$.*

## 3. DDPGPIR Control Design

In this paper, a control strategy for DDPGPIR for the n-link robotic manipulator system with a model for uncertainty and time-varying external disturbances is proposed. The control strategy includes PIR control design, reinforcement learning and policy gradient method, DDPG adaptive PIR control, DDPGPIR network design, the DDPGPIR learning process, and the reward function.

### *3.1. PIR Control Design*

In the n-link robotic manipulator system, the position error $e(t)$ is the difference between the expected joint angle $q_d(t)$ and the actual joint angle $q(t)$. The position error and the error function are defined as follows:

$$e(t) = q_d(t) - q(t) \tag{9}$$

$$s = \dot{e} + \Lambda e \tag{10}$$

where $\Lambda = \Lambda^T = \begin{pmatrix} K_{r1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & K_{rn} \end{pmatrix}$. Take $\dot{q}_s = s(t) + \dot{q}(t)$. Then:

$$\dot{q}_s = \dot{q}_d + \Lambda e \tag{11}$$

$$\ddot{q}_s = \ddot{q}_d + \Lambda e \tag{12}$$

Therefore, the dynamic model Equation (3) of the n-link robotic manipulator can be expressed as follows:

$$\begin{aligned} \tau &= M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + F_f(\dot{q}) + \tau_d \\ &= M(q)(\dot{s} + \ddot{q}) + C(q,\dot{q})(s + \dot{q}) + G(q) - M(q)\dot{s} - C(q,\dot{q})s + F_f(\dot{q}) + \tau_d \\ &= M(q)\ddot{q}_s + C(q,\dot{q})\dot{q}_s + G(q) - M(q)\dot{s} - C(q,\dot{q})s + F_f(\dot{q}) + \tau_d \\ &= E_m - M(q)\dot{s} - C(q,\dot{q})s + E_s \end{aligned} \tag{13}$$

where

$$E_m = M_0(q)\ddot{q}_s + C_0(q,\dot{q})\dot{q}_s + G_0(q) \tag{14}$$

$$E_s = \Delta M(q)\ddot{q}_s + \Delta C(q,\dot{q})\dot{q}_s + \Delta G(q) + F_f(\dot{q}) + \tau_d \tag{15}$$

In PIR control [36], the control law is designed as follows:

$$\tau = \tau_m + \tau_{pi} + v \tag{16}$$

$$\tau_m = M_0(q)\ddot{q}_s + C_0(q,\dot{q})\dot{q}_s + G_0(q) \tag{17}$$

$$\tau_{pi} = K_p s + K_i \int s dt \tag{18}$$

$$v = K_s sgn(s) \tag{19}$$

where $\tau$ is the torque applied to each joint of the n-link robotic manipulator, $\tau_m$ is the torque control term of the model, $K_p$ and $K_i$ are the gain of the proportional term and the gain of the integral term, respectively, and $\tau_s$ is the robust term used to compensate the nonlinear dynamic model error and external disturbance. From Equation (13) and Equation (16), it can be concluded that:

$$M(q)\dot{s} + C(q,\dot{q})s + K_i \int_0^t s dt = -K_p s - K_s sgn(s) + E_s \tag{20}$$

Select the Lyapunov function as follows:

$$V = \frac{1}{2}s^T M s + \frac{1}{2}\left(\int_0^t s d\tau\right)^T K_i \left(\int_0^t s d\tau\right) \tag{21}$$

The derivation on both sides of the equation leads to:

$$\begin{aligned} \dot{V} &= s^T\left[M\dot{s} + \tfrac{1}{2}\dot{M}s + K_i\textstyle\int_0^t s d\tau\right] \\ &= s^T\left[M\dot{s} + Cs + K_i\textstyle\int_0^t s d\tau\right] \\ &= s^T\left[-K_p s - K_s sgn(s) + E_s\right] \\ &= -s^T K_p s - \sum_{i=1}^{n} K_{sii}|s|_i + s^T E \end{aligned} \tag{22}$$

Because of $K_{sii} \geq |E_i|$, $\dot{V} \leq -s^T K_p s \leq 0$. Therefore, the control system is asymptotically stable.

### *3.2. Reinforcement Learning and Policy Gradient Method*

Reinforcement learning is an important branch of machine learning, which is mainly composed of environment, agent, reward, state, and action. When the agent performs action $a_t$ on the environment in state $s_t$, the environment will give the agent a reward $r_{t+1}$, the state changes to the next state $s_{t+1}$, and the future reward value passes through the discount coefficient $\gamma(0 \leq \gamma \leq 1)$ After weighting, the cumulative reward $r_t$ can be expressed as:

$$r_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots \tag{23}$$

The policy of reinforcement learning is the functional relationship $\pi$ between state space and action space. The objective of a policy-based reinforcement learning method is to try to find the optimal strategy $\pi^*$ to maximize the cumulative reward. In the strategy gradient method, the optimal strategy is updated along the gradient direction of the expected cumulative reward as follows:

$$J(\theta) = \mathbb{E}\left(\sum_{l=0}^{N} r(s_l, a_l)|\pi_\theta\right) = \sum_{\sigma} P(\sigma|\theta)r(\sigma) \tag{24}$$

$$\theta_{h+1} = \theta_h + \vartheta \nabla_\theta J(\pi(\theta_h)) \tag{25}$$

where $\theta$ is the parameter vector of the policy, $J(\theta)$ is the objective function of reinforcement learning, $\sigma = (s_0, a_0, s_1, a_1, \cdots s_l, a_l)$ is a group of state action sequences, and $P(\sigma|\theta)$ is the action sequence $\sigma$ Probability of occurrence, $\vartheta$ is the learning rate and $h$ is the number of the current update.

### *3.3. DDPG Adaptive PIR Control*

The schematic diagram of the DDPGPIR control system of the n-link robot manipulator is shown in Figure 1. The input of the controller is the error vector $e = (e_1, e_2, \cdots e_n)$ of the n-link robot manipulator. The output is the torque vector acting on the joint $\tau = (\tau_1, \tau_2, \cdots \tau_n)$. The control performance of the DDPGPIR mainly depends on the parameter vector $g = \left(K_{p1}, K_{i2}, K_{s2}, K_{r2}, \cdots K_{pn}, K_{in}, K_{sn}, K_{rn}\right)$. The control problem of the n-link robot manipulator can be expressed as:

$$\min_{g} \sum_{j=1}^{n} e_j\left(q_{dj}, q_j\left(g_j, p_j\right)\right) \tag{26}$$

where the vector $q_d$ is the expected joint angle, $q$ is the actual joint angle, $p$ is the physical parameter of n-link robot manipulator, and $j$ is the $j$-th link.

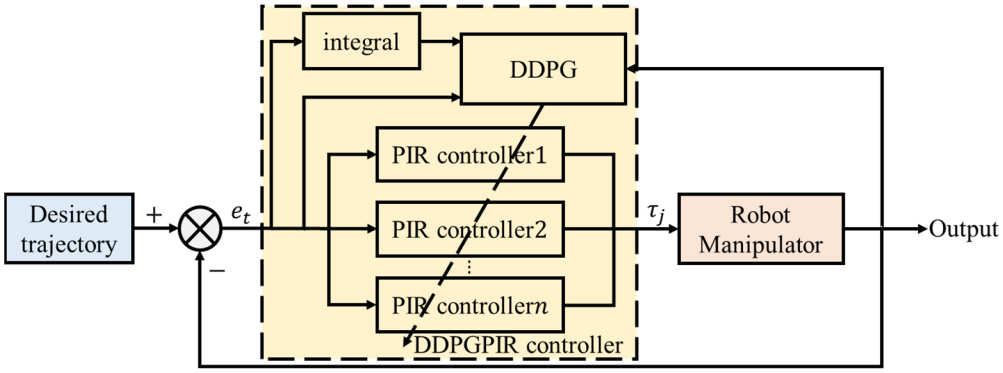


**Figure 1.** Schematic diagram of the DDPGPIR control system.

To improve the adaptability and trajectory tracking accuracy of the n-link robot manipulator, the parameter vector of DDPGPIR needs to be adjusted and optimized in real time. However, the process of setting the parameter is time-consuming, and the optimization process is continuous; it is not advisable to adjust the parameters manually. Therefore, it is necessary to find the optimal strategy function $\mu^*(x)$, which is one of the effective methods for solving this problem. The state vector $x_t = (\tau_1, e_1, \int e_1 dt, \cdots, \tau_n, e_n, \int e_n dt)$ is input into the optimal strategy function to obtain the optimal parameter vector $g_t$. The goal of reinforcement learning is to find the optimal strategy for maximizing cumulative rewards. The objective function can be expressed as:

$$J_\beta(\mu) = \max_{\mu} \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | x_t = x, g_t = \mu(x_t)\right] \tag{27}$$

where $\beta$ is the behavior strategy and $\gamma \in (0, 1)$ is the discount factor.

To find the optimal strategy for maximizing the objective function, the strategy gradient method is usually used to select and execute actions from the distribution function of strategy probability in each time step. However, this method needs to sample continuous actions in each time step, which is a huge calculation process. To solve this problem, the de-

terministic strategy gradient method is used to simplify the calculation process. Therefore, the gradient of the objective function is calculated as follows:

$$\begin{aligned}\nabla_{\theta^\mu} J &\approx \mathbb{E}_{x_t\sim\rho^\beta}\left[\nabla_{\theta^\mu} Q_\mu(x_t,\mu(x_t))\right] \\ &= \mathbb{E}_{x_t\sim\rho^\beta}\left[\nabla_{\theta^\mu} Q(x,g;\theta^Q)\big|_{x=x_t,g=\mu(x_t|\theta^\mu)}\right] \\ &= \mathbb{E}_{x_t\sim\rho^\beta}\left[\nabla_g Q(x,g|\theta^Q)\Big|_{x=x_t,g=\mu(x_t)}\nabla_{\theta^\mu}\mu(x_t|\theta^\mu)\Big|_{x=x_t}\right]\end{aligned} \tag{28}$$

### *3.4. Network Design of DDPGPIR*

The network structure of DDPGPIR includes an actor network, a critic network, and a corresponding target network. The structure of the actor network is shown in Figure 2. The input is the state vector $x_t$ of the n-link robot manipulator, the two middle hidden layers are the full connection layer and the activation layer, and the output layer is the parameter vector $g_t$. The structure of the critic network is shown in Figure 3. The input includes state vector $x_t$ and parameter vector $g_t$. The four middle hidden layers are the full connection layer, activation layer, superposition layer and activation layer. The output layer is the $Q$ value of action.

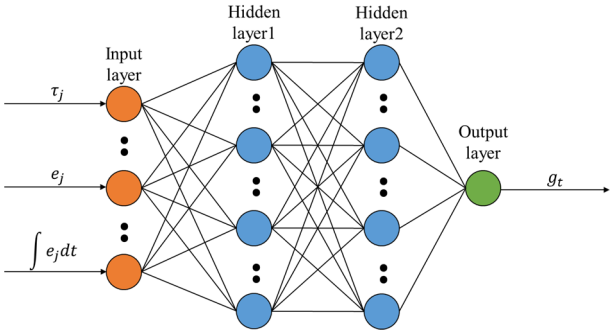


**Figure 2.** The structure of the actor network.

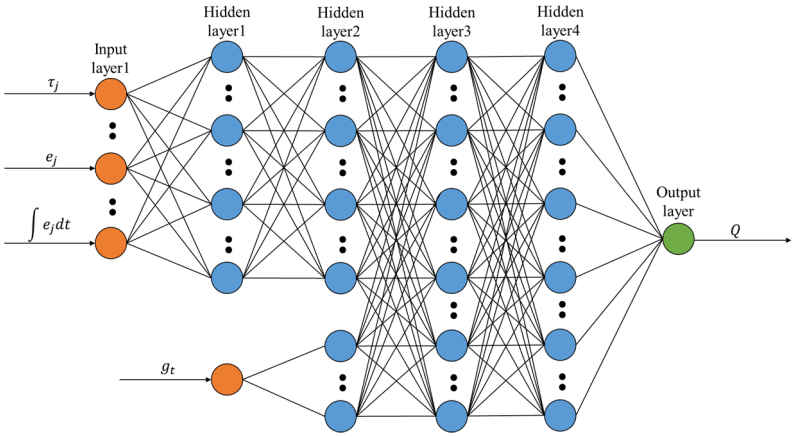


**Figure 3.** The structure of the critic network.

To make the training data relatively independent, to accelerate the convergence speed, and to improve the stability of the network update process, the data used for the current network update are not the previous state data obtained by decision-making, but $M$ small batch sample data randomly selected from the experience replay memory. The critic network includes the current critic network $Q(g,x|\theta^Q)$ and the target critic net-

work $Q'\left(\theta^{Q'}\right)$.The current critic network is updated using a gradient descent method by minimizing the loss function as follows:

$$Q_{target} = r_i + \gamma Q'\left(x_{i+1}, \mu'\left(x_{i+1}\middle|\theta^{\mu'}\right)\middle|\theta^{Q'}\right) \tag{29}$$

$$L = \frac{1}{M}\sum_{i=1}^{M}\left(Q_{target} - Q\left(x_i, g_i\middle|\theta^{Q}\right)\right)^2 \tag{30}$$

$$\nabla L\left(\theta^{Q}\right) = \frac{1}{M}\left[Q_{target} - Q\left(x, g\middle|\theta^{Q}\right)\nabla_{\theta^Q} Q(x, g|\theta^{\mu})\right] \tag{31}$$

where $Q_{target}$ is the value of the target critic network, $Q(x, g|\theta^{\mu})$ is the value of the critic network, $i$ is the $i$th sample data, and $\gamma(0 \le \gamma \le 1)$ is the discount rate. The actor network includes the current actor network $\mu(x|\theta^{\mu})$ and the target actor network $\mu'\left(\theta^{\mu'}\right)$. The current actor network is updated with the deterministic strategy gradient as follows:

$$\nabla_{\theta^\mu} J_\beta(\mu) \approx \frac{1}{M}\sum_i\left(\nabla_g Q\left(x, g\middle|\theta^{Q}\right)\Big|_{x=x_i, g=\mu(x_i)}\nabla_{\theta^\mu}\mu(x;\theta^{\mu})\Big|_{x=x_i}\right) \tag{32}$$

where $\nabla_{\theta^\mu} J_\beta(\mu)$ represents the gradient direction of the $Q$ value caused by the action strategy $\mu$, $\nabla_g Q(x, g|\theta^{Q})|_{x=x_i, g=\mu(x_i)}$ represents the change in the $Q$ value caused by action $\mu(x_i)$ in the current state, and $\nabla_{\theta^\mu}\mu(x;\theta^{\mu})|_{x=x_i}$ is the gradient direction of the current strategy.

The target critic network and the target actor network update the network with a soft update with an update rate of $\rho$ as follows:

$$\begin{cases} {\theta_{i+1}}^{Q'} \leftarrow \rho\theta^{Q} + (1-\rho){\theta_i}^{Q'} \\ {\theta_{i+1}}^{\mu'} \leftarrow \rho\theta^{\mu} + (1-\rho){\theta_i}^{\mu'} \end{cases} \tag{33}$$

### *3.5. Learning Process of DDPGPIR*

The DDPGPIR learning process applied to the manipulator is shown in Figure 4. $\mu(x|\theta^{\mu})$ and $\mu'\left(x\middle|\theta^{\mu'}\right)$ are the current actor network and the target actor network, respectively, and $Q\left(x, g\middle|\theta^{Q}\right)$ and $Q'\left(x, g\middle|\theta^{Q'}\right)$ are the current critic network and the target critic network, respectively. The learning process is described as Algorithm 1. First, parameters $(Q, \mu, Q', \mu')$, memory playback space $RM$, and noise $G$ of the online network and the target network are initialized. After the dynamic information $x_t$ of the manipulator is input into the DDPGPIR agent, according to strategy $\mu$ and noise $G$ to determine the optimal parameter $g_t$ of the PIR controller, the output torque of the controller acts on the manipulator. In addition, the system monitors the joint angle $q_d(t)$ in real time. If $q_d(t)$ is within a reasonable range, the corresponding reward value will be obtained after this action is executed, and the next state $x_{t+1}$ will be input. Otherwise, the action is immediately stopped, a negative reward is given, and the agent re-selects the new action and executes it. The data $(x_t, g_t, r_t, x_{t+1})$ tuple formed in this process will be stored in the experience replay memory $RM$. Small-batch tuple data are randomly extracted from $RM$, the minimal loss function method is used to update the critic network, the deterministic strategy gradient method is used to update the actor network, and the target network is updated by the soft update method.

**Algorithm 1.** DDPGPIR Algorithm.

Initialize the critic network $Q(g, x|\theta^Q)$ and the actor network $\mu(x|\theta^\mu)$
Initialize the target network $Q'\left(\theta^{Q'}\right)$ and $\mu'\left(\theta^{\mu'}\right)$ with the same weights
Initialize replay memory $RM$
Initialize Gaussian noise $G$
for episode = $1 \cdots M$ do
  Receive initial observation state $x_1$
  for $t = 1 \cdots T$ do
    select action $g_t = \left(K_{p1}, K_{i1}, K_{s1}, K_{r1}, \cdots K_{pn}, K_{in}, K_{sn}, K_{rn}\right) = \mu(x_t|\theta^\mu) + G$
    select execution action $g_t$
    if $q(t) \notin [-\varepsilon, \varepsilon]$
      reject $g_t$ and add a negative number to $r$
    else:
      execute $g_t$ and get observed reward $r_t$ and observe new state $x_{t+1}$
    store transition $(x_t, g_t, r_t, x_{t+1})$ in $RM$
    sample mini-batch of $M$ transitions $(x_i, g_i, r_i, x_{t+1})$ from $RM$
    set $Q_{target} = r_i + \gamma Q'\left(x_{i+1}, \mu'\left(x_{i+1}\middle|\theta^{\mu'}\right)\middle|\theta^{Q'}\right)$
    update critic according to Equations (29) and (31)
    update actor according to Equation (32)
    update the target networks according to Equation (33)
  end for
end for

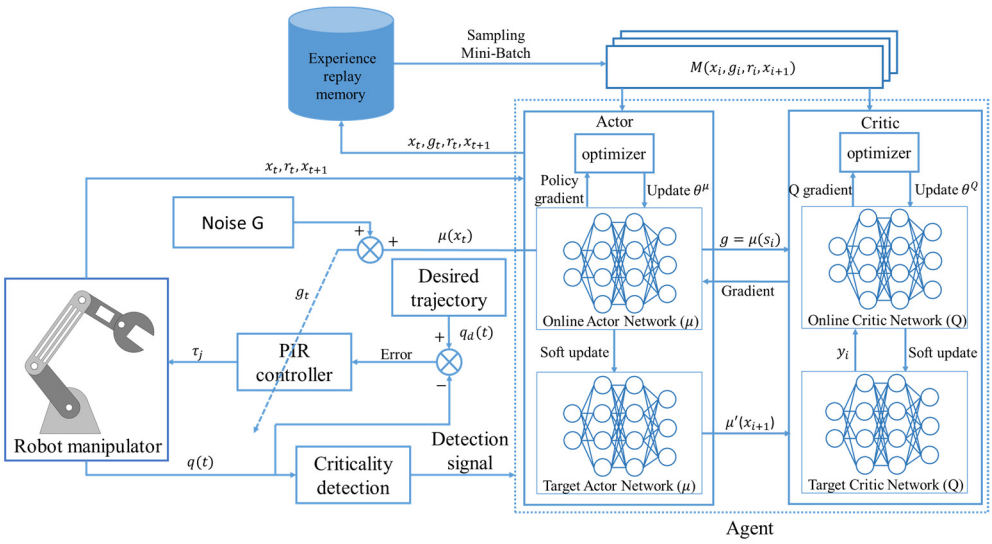


**Figure 4.** DDPGPIR learning process for the manipulator.

### *3.6. Reward Function*

As stated, for most reinforcement learning tasks, there is always a reward function, which can reward each behavior of the agent accordingly, so that the agent can make a corresponding behavior when facing different states and obtain a higher cumulative reward value. To adapt to different reinforcement learning tasks, the reward function must be universal and provide abundant information for the reinforcement learning agents. In the problems discussed in this paper, the trajectory tracking error $e(t)$ and the joint angle $q(t)$ of the manipulator are the variables of most concern. When the tracking error $e(t)$ increases, or the joint angle $q(t)$ exceeds the reasonable range, a negative reward value should be given; otherwise, a positive reward value should be given. Therefore, the reward function combining the Gaussian function and the Euclidean distance is as follows:

$$r = \alpha r_1 + \beta r_2 + \delta r_3 \tag{34}$$

$$r_1 = \sum_{j=1}^{n} -\frac{\left(q_{d_j}(t) - q_j(t)\right)^2}{2c^2} \tag{35}$$

$$r_2 = \sqrt{\sum_{j=1}^{n}\left(q_{d_j}(t) - q_j(t)\right)^2} \tag{36}$$

$$r_3 = \begin{cases} 0, & |q_j(t)| < \varepsilon \\ -1, & other \end{cases} \tag{37}$$

where $\alpha$, $\beta$ and $\delta$ are the coefficients of the reward items, $q_{d_j}(t)$ and $q_j(t)$ are the expected joint angle and the actual joint angle of the $j$-th joint, respectively, and $\varepsilon$ is a reasonable critical value of the joint angle.

## 4. Experiment and Results

To verify the control performance of DDPGPIR, taking a two-link robotic manipulator as an example, the DDPGPIR, PIR and RBFNN are simulated and compared in MATLAB/Simulink. The dynamic model of two joint manipulators can be deduced by the Lagrange method [37,38]. The widely studied kinetic models and parameters can be expressed as follows [39]:

$$M(q) = \begin{bmatrix} p_1 + p_2 + 2p_3\cos q_2 & p_2 + p_3\cos q_2 \\ p_2 + p_3\cos q_2 & p_2 \end{bmatrix} \tag{38}$$

$$C(q,\dot{q}) = \begin{bmatrix} -p_3\dot{q}_2\sin q_2 & -p_3(\dot{q}_1 + \dot{q}_2)\sin q_2 \\ p_3\dot{q}_1\sin q_2 & 0 \end{bmatrix} \tag{39}$$

$$G(q) = \begin{bmatrix} p_4 g\cos q_1 + p_5 g\cos(q_1 + q_2) \\ p_5 g\cos(q_1 + q_2) \end{bmatrix} \tag{40}$$

$$p = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 & p_5 \end{bmatrix}^T = \begin{bmatrix} 2.9 & 0.76 & 0.87 & 3.04 & 0.87 \end{bmatrix}^T \tag{41}$$

In order to achieve better control effect and facilitate comparisons with other control methods, the simulation sampling step size is set at 0.1 s and the simulation cycle is set at 20 s. The initial state of the system is $q_1(0) = -0.5$ rad, $q_2(0) = -0.5$ rad, The expected trajectory path is $q_{d1} = \sin 0.5\pi t$, $q_{d2} = \sin 0.5\pi t$. The friction force is $F_f = 5sgn(\dot{q})$. The external interference is $\tau_d = 10\sin(\dot{q})$. After many attempts, a set of appropriate PIR controller parameters are selected as $K_{p1} = 60$, $K_{i1} = 45$, $K_{s1} = 35$, $K_{r1} = 3$, $K_{p2} = 60$, $K_{i2} = 45$, $K_{s2} = 35$, $K_{r2} = 3$.

RBFNN has good function approximation and generalization ability and is widely used in nonlinear function modeling [40,41]. The adaptive control of manipulators based on RBFNN approximation is as follows [42]:

$$\tau = W^{*T}\varphi(x) + K_v s - v \tag{42}$$

where $W^*$ is the network weight vector and $x$ is the input signal of the network, $\varphi(x)$ is the column vector of the basis function, $K_v$ is the coefficient of error function term, and $v$ is the robust term used to overcome the approximation error of neural network.

### *4.1. Learning Results for DDPGPIR*

In Figure 5, the reward value obtained by the DDPGPIR agent in the initial learning process is low, because the process is in the exploratory stage. However, as the learning times increase, the reward value gradually increases and tends to be stable and close to the expected cumulative reward value, which verifies that the reward function proposed in this paper can effectively avoid the convergence of a deep neural network to the local optimum. At the same time, the correctness and stability of the DDPGPIR model are proved. Figure 6 shows the changing process of the controller parameters. Because the desired trajectory is

constantly changing, the controller parameters are also adjusted in real time, to improve the tracking accuracy of the trajectory.

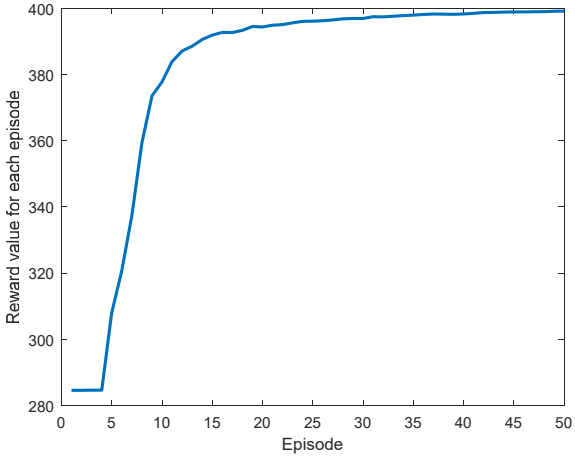


**Figure 5.** Reward value for each episode by DDPGPIR agent.

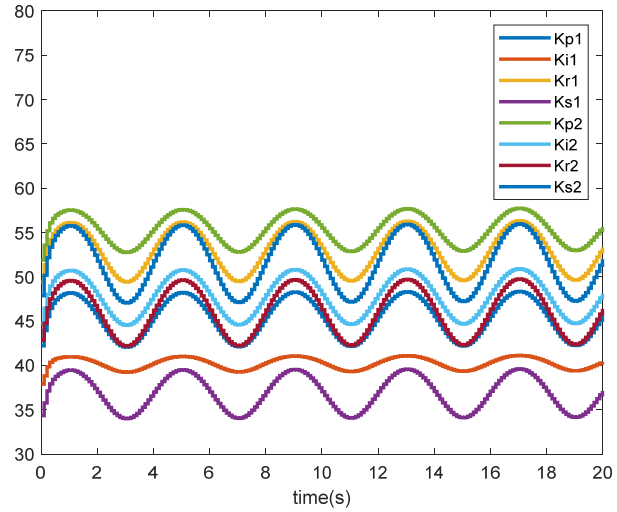


**Figure 6.** Parameter change of the DDPGPIR controller.

### *4.2. Control Effect Comparison of the Controller*

Figures 7–9 show the trajectory tracking performance of the RBFNN, PIR, and DDPG-PIR controllers. The figures show that the DDPGPIR controller has a shorter response time and higher trajectory tracking accuracy than the PIR and RBFNN controllers in the case of friction and time-varying external interference. Figures 10 and 11 show the trajectory tracking errors of the DDPGPIR, PIR, and RBFNN controllers, respectively. It can be seen that compared with DDPGPIR, the PIR and RBFNN controllers have larger overshoot and trajectory tracking errors.

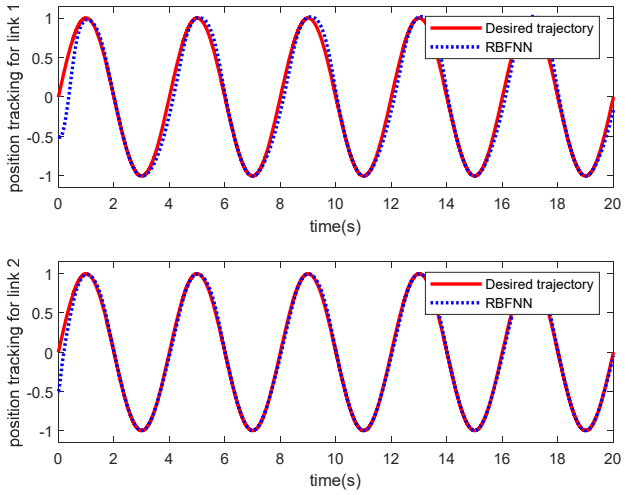


**Figure 7.** Trajectory tracking performance of the RBFNN.

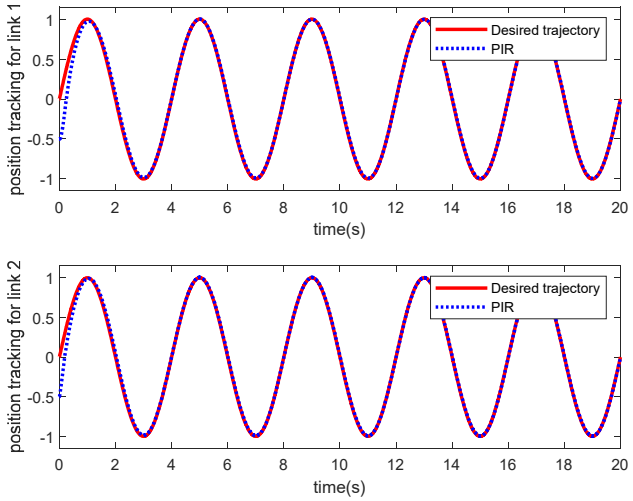


**Figure 8.** Trajectory tracking performance of PIR.

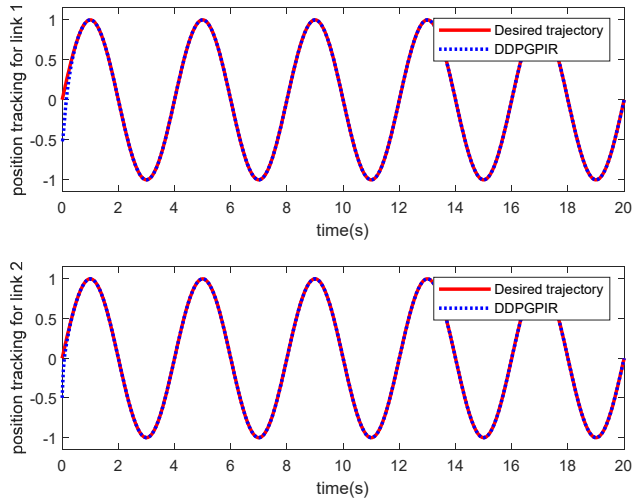


**Figure 9.** Trajectory tracking performance of DDPGPIR.

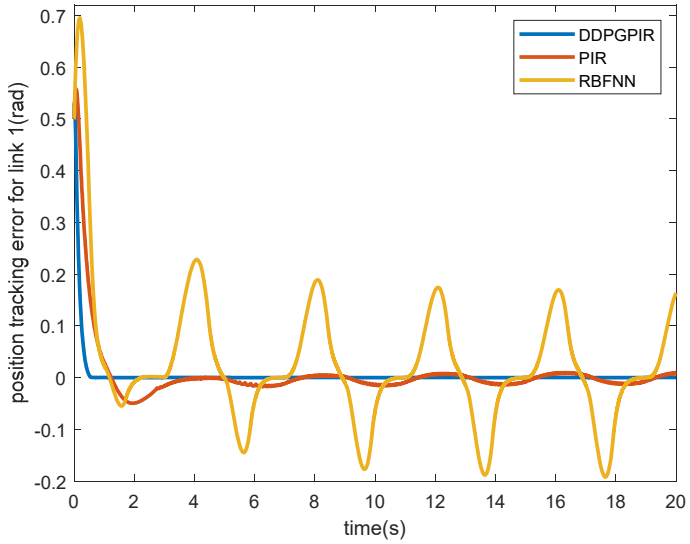


**Figure 10.** Tracking error of joint 1.

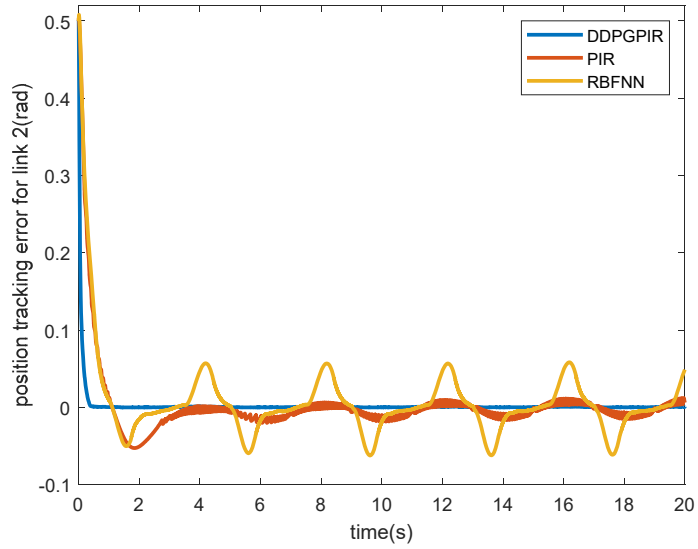


**Figure 11.** Tracking error of joint 2.

### *4.3. Control Performance Index Comparison*

To further highlight the effectiveness of the DDPGPIR controller, the integral absolute error (IAE) and the integral time absolute error (ITAE) were used to evaluate the performance of the controller. The definitions of IAE and ITAE are as follows:

$$\text{IAE} = \int |e| dt \tag{43}$$

$$\text{ITAE} = \int t|e| dt \tag{44}$$

Table 1 shows the IAE and ITAE values of RBFNN, PIR, and DDPGPIR. The table shows that DDPGPIR has smaller IAE and ITAE values than PIR and RBFNN. Therefore, DDPGPIR has better adaptability and robustness in the case of friction and external disturbance.

**Table 1.** Performance index calculation results.

| Controller | Indicator | Joint 1 | Joint 2 |
|---|---|---|---|
| RBFNN | IAE | 1.5978 | 0.5440 |
| | ITAE | 13.4532 | 3.9454 |
| PIR | IAE | 0.4217 | 0.3476 |
| | ITAE | 1.6596 | 1.5451 |
| DDPGPIR | IAE | 0.0866 | 0.0410 |
| | ITAE | 0.0285 | 0.0848 |

## 5. Conclusions

An adaptive PIR control method based on deep reinforcement learning is proposed for the n-link robot manipulator system with model uncertainty and time-varying external disturbances. In this method, the parameters of the PIR controller are adjusted and optimized in real time by using the DDPG algorithm. Among them, the adaptive robust term is used to compensate for the uncertainty of the robot manipulator system. In addition, the model-free reinforcement learning method does not need to rely on expert knowledge and human intervention. The agent of the deep neural network can effectively avoid reduction of control accuracy caused by sparse discretization and the curse of dimension caused by dense discretization. In addition, a reward function combining the Gaussian function and the Euclidean distance is designed to ensure efficient and stable learning of the reinforcement learning agent.

The proposed method was applied to control the two-link robot manipulator with a model for uncertainty and external disturbance. The experimental results show that the reward value obtained by the DDPGPIR agent increases gradually with the increase of learning times, and finally tends to be stable and close to the expected reward value, which proves the correctness and stability of the DDPGPIR model. In addition, compared with PIR and RBFNN, DDPGPIR has better adaptability and robustness, and a higher precision trajectory tracking ability, for the uncertainty of the n-link robot manipulator system. At the same time, it is better than PIR and RBFNN in the performance evaluation of IAE and ITAE.

In future work, since the proposed control method can control the n-link robot arm system, this method may be applied to more complex control tasks, such as unmanned aerial vehicles. However, the ability of the control system to handle emergencies remains a thorny issue. Therefore, our follow-up work will continue to carry out in-depth research for this problem.

**Author Contributions:** Conceptualization, P.L. and J.X.; methodology, P.L.; software, P.L.; validation, W.H. (Wenkai Huang), P.L. and J.X.; formal analysis, P.L.; investigation, F.Z.; resources, W.H. (Wei Hu); data curation, P.L.; writing—original draft preparation, P.L.; writing—review and editing, W.H. (Wenkai Huang); visualization, J.X.; supervision, W.H. (Wenkai Huang); project administration, W.H. (Wenkai Huang); funding acquisition, W.H. (Wenkai Huang). All authors have read and agreed to the published version of the manuscript.

**Funding:** The authors gratefully acknowledge support for this work by the Ministry of Science and Technology of the People's Republic of China under grant nos. 2020AAA0104800 and 2020AAA0104804., as well as Guangzhou Science and Technology Planning Project, grant no. 202002030279.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References


1. Feng, Z.; Hu, G.; Sun, Y.; Soon, J. An overview of collaborative robotic manipulation in multi-robot systems. *Annu. Rev. Control.* **2020**, *49*, 113–127. [CrossRef]
2. Wang, Z.; Cui, W. For safe and compliant interaction: An outlook of soft underwater manipulators. *Proc. Inst. Mech. Eng. Part M J. Eng. Marit. Environ.* **2020**, *235*, 3–14. [CrossRef]
3. Kuo, C.-H.; Dai, J.S.; Dasgupta, P. Kinematic design considerations for minimally invasive surgical robots: An overview. *Int. J. Med Robot. Comput. Assist. Surg.* **2012**, *8*, 127–145. [CrossRef]
4. Albu-Schaffer, A.; Bertleff, W.; Rebele, B.; Schafer, B.; Landzettel, K.; Hirzinger, G. ROKVISS—Robotics component verification on ISS—Current experimental results on parameter identification. In Proceedings of the 2006 IEEE International Conference on Robotics and Automation, Orlando, FL, USA, 15–16 May 2006; pp. 3879–3885.
5. Sage, H.G.; De Mathelin, M.F.; Ostertag, E. Robust control of robot manipulators: A survey. *Int. J. Control.* **1999**, *72*, 1498–1522. [CrossRef]
6. Pan, H.; Xin, M. Nonlinear robust and optimal control of robot manipulators. *Nonlinear Dyn.* **2013**, *76*, 237–254. [CrossRef]
7. Loucif, F.; Kechida, S.; Sebbagh, A. Whale optimizer algorithm to tune PID controller for the trajectory tracking control of robot manipulator. *J. Braz. Soc. Mech. Sci. Eng.* **2019**, *42*, 1. [CrossRef]
8. Elkhateeb, N.A.; Badr, R.I. Novel PID Tracking Controller for 2DOF Robotic Manipulator System Based on Artificial Bee Colony Algorithm. *Electr. Control. Commun. Eng.* **2017**, *13*, 55–62. [CrossRef]
9. Ardeshiri, R.R.; Khooban, M.H.; Noshadi, A.; Vafamand, N.; Rakhshan, M. Robotic manipulator control based on an optimal fractional-order fuzzy PID approach: SiL real-time simulation. *Soft Comput.* **2019**, *24*, 3849–3860. [CrossRef]
10. Ardeshiri, R.R.; Kashani, H.N.; Ahrabi, A.R. Design and simulation of self-tuning fractional order fuzzy PID controller for robotic manipulator. *Int. J. Autom. Control.* **2019**, *13*, 595. [CrossRef]
11. Shah, D.; Chatterjee, S.; Bharati, K.; Chatterjee, S. Tuning of Fractional-Order PID Controller—A Review. In *Frontiers in Computer, Communication and Electrical Engineering*; Taylor & Francis Group: London, UK, 2016; pp. 323–329.
12. Kong, L.; Zhang, S.; Yu, X. Approximate optimal control for an uncertain robot based on adaptive dynamic programming. *Neurocomputing* **2020**, *423*, 308–317. [CrossRef]
13. Kong, L.; He, W.; Dong, Y.; Cheng, L.; Yang, C.; Li, Z. Asymmetric Bounded Neural Control for an Uncertain Robot by State Feedback and Output Feedback. *IEEE Trans. Syst. Man Cybern. Syst.* **2019**, *51*, 1735–1746. [CrossRef]
14. Wang, S. Adaptive Fuzzy Sliding Mode and Robust Tracking Control for Manipulators with Uncertain Dynamics. *Complexity* **2020**, *2020*, 1492615. [CrossRef]
15. Yang, C.; Jiang, Y.; Na, J.; Li, Z.; Cheng, L.; Su, C.-Y. Finite-Time Convergence Adaptive Fuzzy Control for Dual-Arm Robot with Unknown Kinematics and Dynamics. *IEEE Trans. Fuzzy Syst.* **2018**, *27*, 574–588. [CrossRef]
16. Rouhani, E.; Erfanian, A. A Finite-time Adaptive Fuzzy Terminal Sliding Mode Control for Uncertain Nonlinear Systems. *Int. J. Control. Autom. Syst.* **2018**, *16*, 1938–1950. [CrossRef]
17. Yang, Z.; Peng, J.; Liu, Y. Adaptive neural network force tracking impedance control for uncertain robotic manipulator based on nonlinear velocity observer. *Neurocomputing* **2018**, *331*, 263–280. [CrossRef]
18. Guo, Q.; Zhang, Y.; Celler, B.G.; Su, S.W. Neural Adaptive Backstepping Control of a Robotic Manipulator with Prescribed Performance Constraint. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *30*, 3572–3583. [CrossRef] [PubMed]
19. Gosavi, A. Reinforcement Learning: A Tutorial Survey and Recent Advances. *INFORMS J. Comput.* **2009**, *21*, 178–192. [CrossRef]
20. Khan, S.G.; Herrmann, G.; Lewis, F.L.; Pipe, T.; Melhuish, C. Reinforcement learning and optimal adaptive control: An overview and implementation examples. *Annu. Rev. Control.* **2012**, *36*, 42–59. [CrossRef]
21. Liu, C.; Xu, X.; Hu, D. Multiobjective Reinforcement Learning: A Comprehensive Overview. *IEEE Trans. Syst. Man Cybern. Syst.* **2014**, *45*, 385–398. [CrossRef]
22. Kukker, A.; Sharma, R. Stochastic Genetic Algorithm-Assisted Fuzzy Q-Learning for Robotic Manipulators. *Arab. J. Sci. Eng.* **2021**, 1–13. [CrossRef]
23. Runa; Sharma, R.; IEEE. A Lyapunov theory based Adaptive Fuzzy Learning Control for Robotic Manipulator. In Proceedings of the International Conference on Recent Developments in Control, Automation and Power Engineering, Noida, India, 12–13 March 2015; pp. 247–252.
24. Kim, W.; Kim, T.; Kim, H.J.; Kim, S.; IEEE. Three-link Planar Arm Control Using Reinforcement Learning. In Proceedings of the 2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence, Jeju, Korea, 28 June–1 July 2017; pp. 424–428.
25. Du, T.; Cox, M.T.; Perlis, D.; Shamwell, J.; Oates, T.; IEEE. From Robots to Reinforcement Learning. In Proceedings of the 25th International Conference on Tools with Artificial Intelligence, Herndon, VA, USA, 4–6 November 2013; pp. 540–545.
26. Agostinelli, F.; Hocquet, G.; Singh, S.; Baldi, P. From Reinforcement Learning to Deep Reinforcement Learning: An Overview. In *Braverman Readings in Machine Learning: Key Ideas from Inception to Current State*; Rozonoer, L., Mirkin, B., Muchnik, I., Eds.; Springer: Cham, Switzerland, 2018; Volume 11100, pp. 298–328.
27. Wang, H.-N.; Liu, N.; Zhang, Y.-Y.; Feng, D.-W.; Huang, F.; Li, D.-S.; Zhang, Y.-M. Deep reinforcement learning: A survey. *Front. Inf. Technol. Electron. Eng.* **2020**, *21*, 1726–1744. [CrossRef]
28. Mousavi, S.S.; Schukat, M.; Howley, E. Deep Reinforcement Learning: An Overview. In Proceedings of the Sai Intelligent Systems Conference, London, UK, 21–22 September 2016; Volume 16, pp. 426–440.

29. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D.; Continuous Control with Deep Rein-Forcement Learning. *Comput. Sci.* 2015. Available online: https://arxiv.org/abs/1509.02971 (accessed on 18 July 2021).
30. Shi, X.; Li, Y.; Sun, B.; Xu, H.; Yang, C.; Zhu, H. Optimizing zinc electrowinning processes with current switching via Deep Deterministic Policy Gradient learning. *Neurocomputing* **2019**, *380*, 190–200. [CrossRef]
31. Sun, M.; Zhao, W.; Song, G.; Nie, Z.; Han, X.; Liu, Y. DDPG-Based Decision-Making Strategy of Adaptive Cruising for Heavy Vehicles Considering Stability. *IEEE Access* **2020**, *8*, 59225–59246. [CrossRef]
32. Zhao, H.; Zhao, J.; Qiu, J.; Liang, G.; Dong, Z.Y. Cooperative Wind Farm Control with Deep Reinforcement Learning and Knowledge-Assisted Learning. *IEEE Trans. Ind. Inform.* **2020**, *16*, 6912–6921. [CrossRef]
33. Özyer, B. Adaptive fast sliding neural control for robot manipulator. *Turk. J. Electr. Eng. Comput. Sci.* **2020**, *28*, 3154–3167. [CrossRef]
34. Yu, X.; Zhang, S.; Fu, Q.; Xue, C.; Sun, W. Fuzzy Logic Control of an Uncertain Manipulator with Full-State Constraints and Disturbance Observer. *IEEE Access* **2020**, *8*, 24284–24295. [CrossRef]
35. Nohooji, H.R. Constrained neural adaptive PID control for robot manipulators. *J. Frankl. Inst.* **2020**, *357*, 3907–3923. [CrossRef]
36. Ge, S.S.; Lee, T.H.; Harris, C.J. *Adaptive Neural Network Control of Robotic Manipulator*; World Scientific: London, UK, 1998.
37. Zhang, D.; Wei, B. A review on model reference adaptive control of robotic manipulators. *Annu. Rev. Control.* **2017**, *43*, 188–198. [CrossRef]
38. Liu, J.; Dong, X.; Yang, Y.; Chen, H. Trajectory Tracking Control for Uncertain Robot Manipulators with Repetitive Motions in Task Space. *Math. Probl. Eng.* **2021**, *2021*, 8838927. [CrossRef]
39. Xu, W.; Cai, C.; Zou, Y. Neural-network-based robot time-varying force control with uncertain manipulator–environment system. *Trans. Inst. Meas. Control* **2014**, *36*, 999–1009. [CrossRef]
40. Yu, L.; Fei, S.; Huang, J.; Gao, Y. Trajectory Switching Control of Robotic Manipulators Based on RBF Neural Networks. *Circuits Syst. Signal Process.* **2013**, *33*, 1119–1133. [CrossRef]
41. Wang, L.; Chai, T.; Yang, C. Neural-Network-Based Contouring Control for Robotic Manipulators in Operational Space. *IEEE Trans. Control. Syst. Technol.* **2011**, *20*, 1073–1080. [CrossRef]
42. Wang, N.; Wang, D. Adaptive manipulator control based on RBF network approximation. In Proceedings of the 2017 Chinese Automation Congress, Jinan, China, 20–22 October 2017; pp. 2625–2630.