Comparative Study on Control Methods for Two-Degree-of-Freedom Robotic Manipulators

Ruoxuan Huang

College of Engineering, City University of Hong Kong, Hong Kong, China

ruohuang5-c@my.cityu.edu.hk

Abstract. Robotic manipulator systems are inherently nonlinear systems. They usually show model uncertainty and external disturbances. This paper does a systematic comparative analysis of three main control methods for two-degree-of-freedom (2-DOF) robotic manipulators: classical Proportional-Integral-Derivative (PID) control, Model-Based Adaptive Control (MBAC), and intelligent adaptive control based on Radial Basis Function (RBF) neural networks. First, a Lagrangian dynamic model of the 2-DOF robotic manipulator is established. Subsequently, the control laws of the three controllers are elaborated in detail, and system stability is proven through the Lyapunov stability principle. Comparative verification of trajectory tracking is conducted on a simulation platform. The performance differences among the three methods are analyzed in terms of tracking accuracy, convergence speed, disturbance rejection capability, and computational cost. Experimental results indicate that control precision progressively improves from PID to MBAC, and further to RBF neural network control, albeit with a corresponding increase in computational complexity. This study provides a comprehensive performance comparison of classical, model-based, and intelligent control strategies, offering guidance for selecting optimal controllers in nonlinear robotic systems.

Keywords: Robotic manipulator; Dynamics; PID control; Adaptive control; RBF neural network.

1. Introduction

Robotic manipulators, as core components of modern automation systems, have been widely applied in various fields, including manufacturing, aerospace, medical rehabilitation, and even domestic services. Among these applications, high-precision trajectory tracking control is a crucial metric for evaluating their performance and a fundamental basis for accomplishing complex operational tasks. However, a robotic manipulator is inherently a multi-input multi-output (MIMO) system, strongly coupled, and highly nonlinear in its dynamics. Furthermore, in practical operation, it frequently encounters parameter uncertainties (such as payload variations and joint friction) and unknown external disturbances. This makes the design of a controller possessing high precision, high speed, and strong robustness a long-standing and challenging research topic in the field of robot control.

To address these challenges, researchers have proposed various control strategies. In industrial settings, Proportional-Integral-Derivative (PID) controllers are widely utilized due to their simple structure, ease of implementation, and independence from precise system models. Nevertheless, traditional PID controllers employ fixed gains, making them difficult to adapt to the time-varying dynamic characteristics and complex nonlinear coupling of robotic manipulators. Consequently, they often perform poorly in high-precision, high-dynamic tracking tasks, which can potentially lead to significant steady-state errors [1].

To overcome the limitations of traditional PID control, advanced control theories have been introduced into robotic manipulator control. Among these, intelligent control methods, particularly those based on Neural Networks (NNs), have gained significant attention due to their powerful nonlinear approximation and self-learning capabilities. For instance, research by Liu et al. demonstrated that Radial Basis Function (RBF) neural networks can effectively approximate complex nonlinear functions of robotic manipulators. Their adaptive control scheme significantly improved robustness and control precision by reducing the trajectory tracking error range from $[-0.5, 0.5]$

rad with traditional PID control to $[0, 0.2]$ rad. This model-independent characteristic highlights the immense potential of neural network control in handling highly uncertain systems [2]. Besides, Long et al. proposed an adaptive robust self-tuning PID strategy. Through online update algorithms, it enabled PID gains to autonomously respond to time-varying uncertainties and abrupt faults. In terms of trajectory tracking performance, its normalized mean square error (NMSE) reached 28.2×10^{-3}, outperforming that 29.5×10^{-3} of the hybrid adaptive PID (HPID), thereby significantly enhancing the adaptability and robustness of PID controllers [3]. Also, Hamed Rahimi Nohooji also introduced a constrained neural adaptive PID controller. It estimated uncertainties and directly determined PID gains using RBF neural networks. Numerical simulations have verified that this method can effectively confine errors within predefined constraints [1]. Additionally, other model-free approaches have been explored. For example, Huang et al. utilized Time Delay Estimation (TDE) to manage system uncertainties and combined with a novel performance function. They achieved tracking control with a specified precision of ± 0.005 within a preset time of 1.5s. This provided new insights into controller performance guarantees [4]. Yu Shiwei et al., focused on upper limb exoskeleton rehabilitation robots and designed an RBF neural network adaptive control method with a sliding mode robust term. It approximated system uncertainties, optimized driving torques and achieved precise trajectory tracking. Simulation results showed that its trajectory tracking accuracy improved by 66.87% and 27.57% compared to traditional computed torque control [5, 6].

Based on the aforementioned research, this paper will focus on investigating three strategies: classic PID control, model-based adaptive control (MBAC) and RBF neural network-compensated adaptive control. Comparative simulation experiments will be conducted using the MATLAB/Simulink platform to analyze the characteristics of these three methods in terms of trajectory tracking accuracy, convergence properties, and computational cost. Through comparison, the aim is to reveal the intrinsic connections between different control strategies and to select and design the optimal robotic manipulator controller for varying performance requirements and degrees of model knowledge.

2. Research Methods

2.1. Robotic Manipulator Dynamics Model and Control Objectives

2.1.1 N-link robotic manipulator dynamics

This paper employs the Lagrangian formalism to establish the dynamic model of the robotic manipulator, obtaining the relationship between forces or torques and the manipulator's angular displacement, angular velocity, and angular acceleration. Its dynamic model can be described by a second-order nonlinear differential equation as [6]:

$$M(q)\ddot{q} + Vm(q, \dot{q})\dot{q} + F_d(\dot{q}) + G(q) = \tau \tag{1}$$

where $q, \dot{q}, \ddot{q} \in R^n$ are the joint angle, velocity, and acceleration vectors, $M(q) \in R^{n \times n}$ is the inertia matrix, $Vm(q, \dot{q})\dot{q} \in R^{n \times n}$ is the Coriolis and centrifugal force matrix, $G(q) \in R^n$ is the gravity term vector, $F_d(\dot{q}) \in R^n$ is the friction force term vector, and $\tau \in R^n$ is the control torque input vector.

2.1.2 Dynamic properties

For all configurations q, the inertia matrix $M(q)$ satisfies:

$$\lambda min(M)I \leqslant M(q) \leqslant \lambda max(M)I \tag{2}$$

Boundedness ensures the physical rationality of the system so that a given torque necessarily produces a definite acceleration, meaning that there is always a solution for $\ddot{q} = M^{-1}(q)\tau_{net}$.
The matrix is skew-symmetric:

$$\zeta^T \left[\dot{M} - 2V_m\right]\zeta = 0, \forall \zeta \in R^n \tag{3}$$

Skew-symmetry reflects the energy conservation property of mechanical systems, and is crucial for designing energy-based Lyapunov function stability analysis.

The dynamic equation can be rewritten into a parametric linear form:

$$M(q)\ddot{q} + Vm(q,\ \dot{q})\dot{q}\ + F(\dot{q}) + G(q) = Y(q,\dot{q},\ddot{q})\Theta \tag{4}$$

where Y is the regression matrix.

2.1.3 Control objectives

Given a desired joint trajectory $q_d(t)$, along with its derivatives $\dot{q}_d(t)$ and $\ddot{q}_d(t)$, the control objective is to design a control torque τ such that the actual joint angle $q(t)$ can precisely track the desired trajectory. The tracking error is defined as:

$$e(t) = q_d(t) - q(t) \tag{5}$$

2.2. Control Method Theory

2.2.1 Proportional-Integral-Derivative (PID) control

PID control is a classic feedback control method that does not rely on a precise system model. For multi-input multi-output (MIMO) robotic manipulator systems, a decoupled PID control strategy is typically adopted. The control law is:

$$\tau(t) = K_P e(t) + K_I \int_0^t e(\tau)d\tau + K_D \dot{e}(t) \tag{6}$$

where K_P, K_I, K_D is a diagonal positive-definite gain matrix. Substituting the control law into dynamic equation (1) yields the closed-loop error dynamics:

$$M(q)\ddot{e} + Vm(q,\ \dot{q})\dot{e} + K_D\dot{e} + K_P e + K_I \int e d\tau = \eta(t) \tag{7}$$

where $\eta(t)$ includes complex nonlinear and coupling terms of the system dynamics.

PID is simple to implement and suppresses uncertainties through feedback. However, theoretically, it is difficult to guarantee global asymptotic stability, and its performance in high-dynamic tasks is limited by its fixed-gain structure.

2.2.2 Model-Based Adaptive Control (MBAC)

MBAC utilizes the known structure of system dynamics (LIP property) but assumes that physical parameters are unknown. Precise control is achieved by estimating parameters.
The filtered tracking error $r(t)$ is defined as [7]:

$$r(t) = \dot{e}(t) + \Lambda e(t) \tag{8}$$

where Λ is a positive-definite diagonal matrix that determines the error convergence speed. The reference velocity $\dot{q}_r$ and reference acceleration $\ddot{q}_r$ are defined as:

$$\dot{q}_r = \dot{q}_d + \Lambda e, \ddot{q}_r = \ddot{q}_d + \Lambda\dot{e} \tag{9}$$

Utilizing the LIP property, a regression matrix $Y_r(q,\dot{q},\dot{q}_r,\ddot{q}_r)$ is constructed such that:

$$M(q)\ddot{q}+Vm(q,\ \dot{q})\dot{q}\ +F_d(\dot{q})+G(q)=Y_r(\cdot)\Theta \tag{10}$$

The control law is:

$$\tau=Y_r(\cdot)\hat{\Theta}(t)+K_D r(t) \tag{11}$$

Substituting the control law into the system dynamics yields the closed-loop error dynamics:

$$M(q)\dot{r}+Vm(q,\ \dot{q})r\ +K_D r\ =Y_r(\cdot)\tilde{\Theta}(t) \tag{12}$$

where $\tilde{\Theta}(t)=\Theta-\hat{\Theta}(t)$ is the parameter estimation error.

A Lyapunov function is constructed as:

$$V(t)=\frac{1}{2}r^T M(q)r+\frac{1}{2}\tilde{\Theta}^T\Gamma^{-1}\tilde{\Theta} \tag{13}$$

where Γ is a positive-definite adaptive gain matrix.

Differentiating $V(t)$ with respect to time, and utilizing the skew-symmetry property of $\left(r^T\left(\frac{1}{2}\dot{M}-V_m\right)r=0\right)$:

$$\dot{V}(t)=-r^T K_D r+\tilde{\Theta}^T\left(\Gamma^{-1}\dot{\tilde{\Theta}}+Y_r^T r\right) \tag{14}$$

To ensure $\dot{V}(t)\leqslant 0$, guarantee system stability and error convergence, the adaptive law is chosen as:

$$\dot{\hat{\Theta}}(t)=\Gamma Y_r^T(\cdot)r(t) \tag{15}$$

Theoretically, MBAC guarantees global asymptotic stability$(e(t)\to 0)$. Its performance relies on the accuracy of the system's structural model and offers high computational efficiency.

2.2.3 RBF Neural Network Adaptive Control (RBF-NN)

RBF-NN control utilizes the universal approximation capability of neural networks to learn and compensate for unknown nonlinear system dynamics, without requiring knowledge of the dynamic structure. The RBF network is used to approximate unknown nonlinear functions, treating the dynamics as a function to be approximated [5-7]:

$$f(x)=M(q)\ddot{q}_r+V_m(q,\dot{q})\dot{q}_r+G(q)+F_d(\dot{q}) \tag{16}$$

The input vector is selected as $x=[q^T,\dot{q}^T,q_d^T,\dot{q}_d^T,\ddot{q}_d^T]^T$.

According to the universal approximation theorem, there exist optimal weight W^* such that $f(x)=W^{*T}H(x)+\epsilon(x)$, where ϵ is the approximation error. $H(x)$ is the Gaussian basis function vector:

$$H_j(x)=\exp\left(-\frac{\|x-c_j\|^2}{2b_j^{\ 2}}\right) \tag{17}$$

The control law is:

$$\tau = \hat{W}^T H(x) + K_D r \tag{18}$$

where $\hat{W}$ is the estimated value of the network weights.

The closed-loop error dynamics become:

$$M(q)\dot{r} + Vm(q,\ \dot{q})r\ + K_D r\ = \hat{W}^T H(x) + \epsilon(x) \tag{19}$$

where $\tilde{W} = W^* - \hat{W}$ is the weight estimation error.

Lyapunov function is constructed:

$$V(t) = \frac{1}{2} r^T M(q) r + \frac{1}{2} tr\left(\tilde{W}^T \Gamma^{-1} \tilde{W}\right) \tag{20}$$

Differentiating with respect to time and choosing the weight update law to guarantee stability:

$$\dot{\hat{W}} = \Gamma H(x) r^T \tag{21}$$

RBF-NN control does not rely on a dynamic model, offers strong adaptability, and theoretically guarantees uniform ultimate boundedness of errors. However, its computational complexity is significantly higher than PID control and MBAC control.

2.3. Lumped Parameter Identification for Two-Degree-of-Freedom Robotic Manipulator Dynamics

Fig. 1 illustrates a 2-degree-of-freedom (2-DOF) robotic manipulator, featuring two links and two revolute joints. The joint angles are represented by q_1 and q_2, with corresponding torques τ_1 and τ_2.

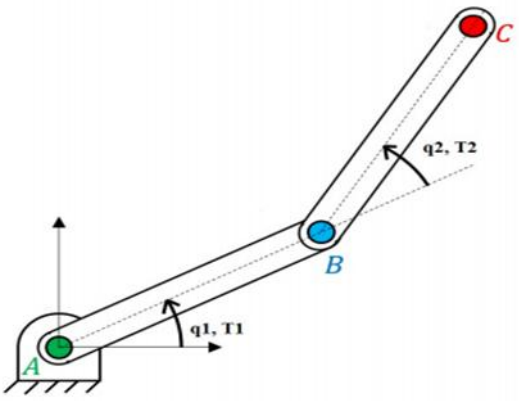

Fig. 1 Schematic Fig. of two-link robotic manipulator (Photo/Picture credit: Original).

The dynamic equations for a two-link system have the following form:

$$\begin{aligned} \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} &= \underbrace{\begin{bmatrix} p_1 + 2p_3\cos q_2 & p_2 + p_3\cos q_2 \\ p_2 + p_3\cos q_2 & p_2 \end{bmatrix}}_{M(q)} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} \\ &+ \underbrace{\begin{bmatrix} -p_3\sin q_2 \cdot \dot{q}_2 & -p_3\sin q_2 (\dot{q}_1 + \dot{q}_2) \\ p_3\sin q_2 \cdot q_1 & 0 \end{bmatrix}}_{V_m(q,\dot{q})} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} + \underbrace{\begin{bmatrix} f_{d1} & 0 \\ 0 & f_{d2} \end{bmatrix}}_{F_d} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} \end{aligned} \tag{22}$$

where $p_2 = m_2 l_{c_2}^2 + I_2$, $p_3 = m_2 l_1 l_{c_2}$, $p_1 = m_1 l_{c_1}^2 + m_2 (l_1^2 + l_2^2) + I_1 + I_2$

Utilizing parameter linearization of the dynamic equations:

$$Y(q,\dot{q},\ddot{q})\Theta = \tau \tag{23}$$

where $\Theta = [p_1, p_2, p_3, f_{d1}, f_{d2}]^T$.

Excitation trajectories are designed in MATLAB to ensure that continuous excitation effectively captures the dynamic characteristics of the 2-DOF robotic manipulator. These trajectories are generated by superimposing multiple harmonic sinusoidal signals [8, 9]. Each joint's trajectory consists of four sinusoidal components with different frequencies (0.5–3.0 Hz) and amplitudes, controlled within reasonable joint limits (≤ 0.3 rad) to avoid excessive torque. The trajectory is defined as:

$$q_{\exp}(t) = \sum_{k=1}^{4} A_k \sin(2\pi f_k t) \tag{24}$$

where $f_k = [0.5, 1.0, 2.0, 3.0], A_k = [0.3, 0.2, 0.15, 0.1]$.

Record the applied torque τ and the response $(q,\dot{q},\ddot{q})$. Through constructing a regression matrix, the solution is obtained by using the least squares method:

$$\hat{\Theta} = (Y^T Y)^{-1} Y^T \tau_{measured} \tag{25}$$

The identified parameters are: $p_1 = 3.473, p_2 = 0.196, p_3 = 0.242, f_{d1} = 5.3, f_{d2} = 1.1$

3. MATLAB/Simulink Simulation Comparative Verification

The experiment is based on the parameterized two-degree-of-freedom robotic manipulator model obtained from parameter identification: $\Theta_{true} = [p_1, p_2, p_3, f_{d1}, f_{d2}]^T = [3.473, 0.196, 0.242, 5.3, 1.1]^T$

3.1.Control Task and Trajectory

The robotic manipulator is required to track a time-varying trajectory (duration of 30 seconds):

$$q_d(t) = \begin{bmatrix} 0.5\sin(t) \\ 2\cos\left(\frac{t}{4}\right) \end{bmatrix} \tag{26}$$

The initial state is set with an initial error $\Delta q(0) = [0.1, -0.2]^T$ to evaluate convergence performance.

3.2.Controller Parameter Configuration

Table 1 shows the parameter configuration.

Table 1. The parameter configuration.

Methods	Key Parameter	Value
PID	K_p, K_I, K_D	$\mathrm{diag}(500), \mathrm{diag}(100), \mathrm{diag}(100)$
MBAC	Feedback Gain K_D	$\mathrm{diag}(30)$
	Filter Gain Λ	$\mathrm{diag}(10)$
	Adaptive Gain Γ	$\begin{vmatrix} 5 & 0 & 0 & 0 & 1 \\ 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 3 & 5 & 0 \\ 0 & 0 & 5 & 20 & 0 \\ 1 & 0 & 0 & 0 & 10 \end{vmatrix}$
RBF-NN	Feedback Gain K_D	$\mathrm{diag}(50)$
	Filter Gain Λ	$\mathrm{diag}(20)$
	Learning Rate Γ	800
	Number of Neurons N	150

Root Mean Square Error (RMSE) is used as a quantitative evaluation metric to measure the average tracking accuracy over the entire simulation period:

$$RMSE_{total} = \sqrt{\frac{1}{T}\int_0^T \| e(t) \|^2 dt} \tag{27}$$

3.3.Simulation Results Analysis

3.3.1 Performance Metric Comparison (RMSE)

Fig. 2 shows the performance comparison of total RMSE, where lower values are better. Table 2 also illustrates the RMSE comparison, including that of joint1 and joint2.

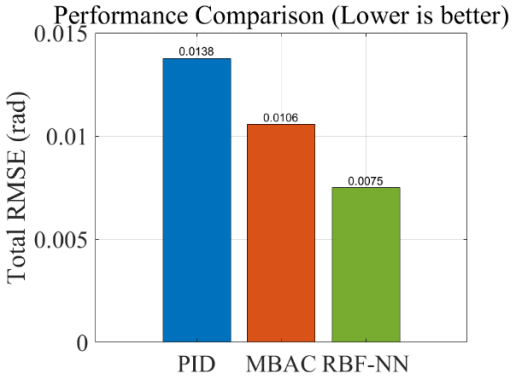

Fig. 2 Total RMSE comparison (Photo/Picture credit: Original).

Table 2. RMSE comparison

Control Methods	Total RMSE	Joint1RMSE	Joint2RMSE
PID	0.01377	0.00693	0.01190
MBAC	0.01057	0.00564	0.00894
RBF-NN	0.00750	0.00401	0.00634

Simulation results show RBF-NN achieved the best tracking accuracy, improving by approximately 45.6% over PID and 29.0% over MBAC. MBAC's tracking accuracy was also superior to PID.

3.3.2 Trajectory tracking performance comparison

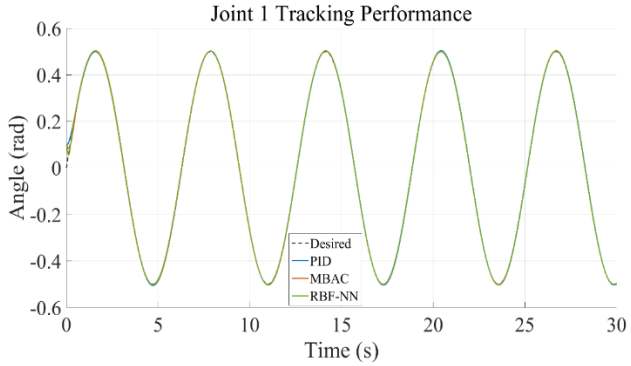

Fig. 3 Joint1 tracking performance (Photo/Picture credit: Original).

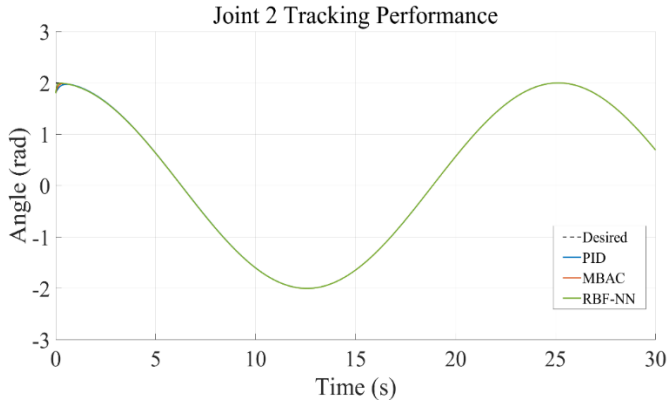

Fig. 4 Joint2 tracking performance (Photo/Picture credit: Original).

All three methods demonstrate effective tracking of the desired trajectory. However, in high-dynamic regions, PID control exhibits relatively noticeable lag, while the other two adaptive control methods show superior dynamic response capabilities (Figs. 3, 4).

3.3.3 Error norm convergence comparison

Fig. 5 shows the comparison between three control methods' tracking error norms.

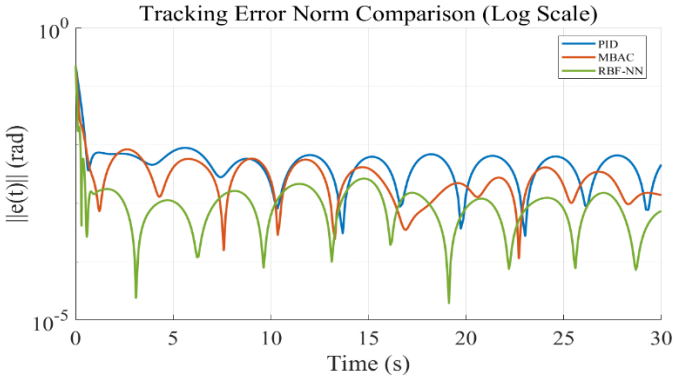

Fig. 5 Comparison of tracking error norms (Photo/Picture credit: Original).

As shown in the Fig. 5, RBF-NN exhibits the fastest convergence speed, rapidly reducing the error norm to a steady-state level within approximately 1-2 seconds. MBAC is the second fastest, while PID is the slowest. In the steady-state phase ($t > 5s$), RBF-NN maintains the lowest error level. The steady-state error of PID is the largest and exhibits significant periodic fluctuations, indicating its limited capability to compensate for the system's periodic nonlinear dynamics. The steady-state error of MBAC lies between the two.

3.3.4 Computational cost comparison

Table 3 presents a comparison of computational costs.

Table 3. Computational cost comparison

Control Methods	Simulation Time(s)
PID	0.041
MBAC	0.047
RBF-NN	0.968

As shown in Table 3, PID and MBAC demonstrate extremely high computational efficiency. In contrast, the computational cost of RBF-NN is significantly higher, approximately 20 times greater than the other two methods. This can be attributed to its high-dimensional state vector and the complexity of its network computations.

4. Conclusion

This study presents a comprehensive comparative analysis of three control strategies—PID, MBAC, and RBF-NN—for trajectory tracking of a two-link robotic manipulator.

RBF-NN demonstrates the best performance in terms of tracking accuracy and dynamic performance. It achieves the highest tracking precision and the fastest convergence rate. This verifies the effectiveness for neural network-based approaches to handle complex nonlinear systems. Additionally, the RBF-NN achieves the minimal steady-state error among all three controllers. On the other hand, the PID controller showed the largest steady-state error and came with pronounced periodic oscillations. This reveals its inherent limitations in compensating for the system's nonlinear dynamics. The error performance of MBAC was intermediate, falling between those of the two methods. But the superior performance of RBF-NN comes at a cost, which has a higher computational cost.

The three controllers differ significantly in terms of model dependency. PID is entirely model-free, which requires no prior knowledge of system dynamics. MBAC requires knowledge of the dynamic model structure to utilize system information. RBF-NN is theoretically model-independent, offering the greatest adaptability to uncertain or unknown system dynamics.

Therefore, in practical robotic applications, the selection of an appropriate control strategy requires a trade-off based on the specific accuracy requirements, available computational resources and the degree of knowledge about the system model.

References

[1] Nohooji H R. Constrained neural adaptive PID control for robot manipulators. Journal of the Franklin Institute, 2020, 357(7): 3907–3923.

[2] Liu H, Zhao T, Song T, Liu Z. Adaptive control of manipulator based on neural network. Neural Computing and Applications, 2021, 33: 4077–4085.

[3] Long M T, Nan W Y, Quan N V. Adaptive robust self-tuning PID fault-tolerant control for robot manipulators. International Journal of Dynamics and Control, 2024, 12: 477–485.

[4] Huang X W, Dong Z Y, Yang P, Zhang L H. Model-free adaptive trajectory tracking control of robotic manipulators with practical prescribed-time performance. Nonlinear Dynamics, 2023, 111: 20015–20039.

[5] Yu S, Lu S, Li Z, et al. Adaptive control method for upper limb exoskeleton rehabilitation robot based on RBF neural network. Computer Era, 2023(10): 83–88.

[6] Yu S. Research on adaptive stable control method for upper limb exoskeleton rehabilitation robot. Master's Dissertation, Shandong Jianzhu University, 2023.

[7] Liu J. RBF Neural Network Adaptive Control MATLAB Simulation. Beijing: Tsinghua University Press, 2014.

[8] Yang Z, Yu J, Yu Z, et al. Research on excitation trajectory optimization for dynamic parameter identification. Journal of Mechanical Transmission, 2024, 48(11): 37–47.

[9] Wang S, Wu C, Liu Y, et al. Research on an improved dynamic parameter identification method for industrial robotic manipulators. Mechanical Design and Manufacturing, 2024(11): 358–361+365.