





Article

Trajectory Planning of Robot Manipulator Based on RBF Neural Network

Qisong Song ¹, Shaobo Li ^{1,2,*}, Qiang Bai ¹, Jing Yang ^{1,2}, Ansi Zhang ^{1,2}, Xingxing Zhang ²
and Longxuan Zhe ¹

- ¹ College of Mechanical Engineering, Guizhou University, Guiyang 550025, China; gs.qsong18@gzu.edu.cn (Q.S.); cme.qbai18@gzu.edu.cn (Q.B.); jyang23@gzu.edu.cn (J.Y.); zhangas@gzu.edu.cn (A.Z.); longxuanzhe18@163.com (L.Z.)
- ² State Key Laboratory of Public Big Data, Guizhou University, Guiyang 550025, China; xingxingzhang_star@163.com
- * Correspondence: lishaobo@gzu.edu.cn

Abstract: Robot manipulator trajectory planning is one of the core robot technologies, and the design of controllers can improve the trajectory accuracy of manipulators. However, most of the controllers designed at this stage have not been able to effectively solve the nonlinearity and uncertainty problems of the high degree of freedom manipulators. In order to overcome these problems and improve the trajectory performance of the high degree of freedom manipulators, a manipulator trajectory planning method based on a radial basis function (RBF) neural network is proposed in this work. Firstly, a 6-DOF robot experimental platform was designed and built. Secondly, the overall manipulator trajectory planning framework was designed, which included manipulator kinematics and dynamics and a quintic polynomial interpolation algorithm. Then, an adaptive robust controller based on an RBF neural network was designed to deal with the nonlinearity and uncertainty problems, and Lyapunov theory was used to ensure the stability of the manipulator control system and the convergence of the tracking error. Finally, to test the method, a simulation and experiment were carried out. The simulation results showed that the proposed method improved the response and tracking performance to a certain extent, reduced the adjustment time and chattering, and ensured the smooth operation of the manipulator in the course of trajectory planning. The experimental results verified the effectiveness and feasibility of the method proposed in this paper.

Keywords: robot manipulator; trajectory planning; trajectory tracking; RBF neural network; adaptive robust controller; modeling



Citation: Song, Q.; Li, S.; Bai, Q.; Yang, J.; Zhang, A.; Zhang, X.; Zhe, L. Trajectory Planning of Robot Manipulator Based on RBF Neural Network. *Entropy* **2021**, *23*, 1207. <https://doi.org/10.3390/e23091207>

Academic Editor: António M. Lopes

Received: 25 July 2021

Accepted: 11 September 2021

Published: 13 September 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

With the advancements in automation and robot technology, robots have begun to be widely used in the industrial, agricultural, and medical fields, among many others. Improving the trajectory planning of robot manipulators is one of the core focuses of robot research, and has great research prospects [1]. Precise robot manipulator trajectories can improve the efficiency of a robot's various tasks, such as workshop operations, crop picking, medical surgery and so on.

A robot manipulator is a nonlinear and uncertain system. Manipulator trajectory planning should not only consider obstacle avoidance, trajectory accuracy, smooth operation, energy consumption, among other factors, but also needs to consider the problems of external interference, communication delay, and the nonlinearity and uncertainty of robot manipulators [2–5]. In order to solve these problems, many researchers have studied the kinematics formula, dynamic model, and control technology of robot manipulators. At present, research into the kinematics formula and dynamic model of robot manipulators has been gradually growing. Research into control technology has mainly focused on the

sliding mode control, robust control and adaptive control, and the nonlinear and uncertain problems can be alleviated by designing controllers [6,7].

However, at present, the design of manipulator controllers is mostly based on low degree of freedom robots, and the communication delay, instability, nonlinearity and uncertainty of the high degree of freedom manipulators have not been effectively solved. These have become difficult and contentious points in current research into manipulator trajectory planning.

This study aims to promote the further development of trajectory planning research, improve the accuracy of manipulator trajectory planning, effectively deal with the nonlinearity and uncertainty of the high degree of freedom manipulators, enable manipulators to obtain good trajectory tracking performance, and better provide corresponding technical guidance for the actual trajectories of manipulators. We carried out the research on the controller design and trajectory planning of a 6-DOF robot.

A trajectory planning method for robot manipulators based on an RBF neural network is proposed in this study, which has the following contributions:

- (1) The study proposes a trajectory planning method for a 6-DOF manipulator, which improves its trajectory tracking performance and motion stability, gives it higher versatility, and can be applied to the trajectory planning of a low DOF manipulator.
- (2) The study designs a new adaptive robust controller based on an RBF neural network, which uses the strong robustness of adaptive control theory and the self-learning and nonlinear characteristics of RBF neural networks to deal with the nonlinearity and uncertainty of high DOF manipulators.
- (3) The study designs and builds an experimental platform for the trajectory planning of manipulators and carries out the actual trajectory planning experiments based on this experimental platform, which verifies the effectiveness and feasibility of the proposed method.

The rest of this paper is organized as follows: Section 2 describes the related work on optimal trajectory planning and robot control methods. Section 3 covers the design of the trajectory planning experimental platform and introduces the overall framework of the trajectory planning method. Section 4 designs a new adaptive controller based on an RBF neural network and uses a Lyapunov function to analyze its stability and convergence. Section 5 presents the simulation and experimental results and analyzes and discusses the results. Section 6 concludes the paper and offers recommendations for future works.

2. Related Work

Robot manipulator trajectory planning takes the ideal trajectory kinematics parameters and the robot manipulator system as the input, and takes the displacement, velocity and acceleration of each joint and end effector as the output. The intermediate point pose is usually solved by linear interpolation [8], polynomial interpolation [9], and other interpolation algorithms.

According to differences in the planning space, trajectory planning can be divided into Cartesian space trajectory planning and joint space trajectory planning. They must both meet the kinematic and dynamic constraints of the robot manipulator, and the trajectory must be continuous, smooth, and impact-free within the performance requirements of the robot manipulator's components; that is, the speed and acceleration must not have sudden changes [10]. At present, the research on optimal trajectory planning mainly focuses on time-optimal trajectory planning, energy-optimal trajectory planning, impact-optimal trajectory planning, and hybrid optimal trajectory planning [11–13].

Time-optimal trajectory planning has high work efficiency [14]. Yi Fang et al. [15] proposed a smooth and time-optimal S-curve trajectory planning method to improve the planning efficiency of manipulators. Kim et al. [16] used trapezoidal velocity curves to quickly approximate the ideal trajectory, so as to approach time-optimal planning dynamically. Zhang et al. [17] proposed an adaptive cuckoo search algorithm with faster convergence speed and higher accuracy to minimize the total motion time.

Energy-optimal trajectory planning is suitable for robots with limited energy storage, such as space exploration robots, underwater robots and military robots [18]. Liu et al. [19] used screw theory and Kane's equations to establish kinematic and dynamic models to achieve energy optimization under the continuous motion of the manipulator. Bakshi et al. [20] optimized the robot path trajectory in multi-task environments, saving about 5–10% in energy consumption while ensuring the same work efficiency.

Impact-optimal trajectory planning aims to optimize the acceleration of each joint of the manipulator [21]. Ma et al. [22] proposed a new convex optimization method, which transforms non-convex jerk into linear acceleration and solves the acceleration limitation problem. Dai et al. [23] used a greedy algorithm to optimize the path of a robot with large jitters during manufacturing tasks, so as to improve its trajectory acceleration performance.

Hybrid-optimal trajectory planning optimizes the trajectory of the manipulator by considering time, energy consumption, impact, and other factors, and this method includes time-energy optimal, time-impact optimal, and time-impact-acceleration optimal trajectory planning [24]. Chen et al. [25] proposed an improved immune clonal selection algorithm to solve multi-objective trajectory planning. Yin et al. [26] proposed a trajectory planning method based on machine learning to generate time energy consumption optimal trajectories. Zhang et al. [27] proposed an improved dolphin swarm algorithm to generate better localization performance and more energy-efficient trajectories.

Although the above studies were able to optimize the trajectories of manipulators, they did not consider the design of the controller, failed to improve the trajectory accuracy, and did not feature the trajectory tracking error.

In [28], a robust controller based on an RBF neural network was designed to improve the trajectory tracking performance of a 3-DOF robot manipulator. In [29], an adaptive controller based on an RBF neural network was designed to solve the dynamic deviation problem of a 2-DOF robot manipulator. In [30], a sliding mode controller was designed to shorten the circular trajectory error of the 3-DOF robot manipulator. In [31], the researchers proposed a robust noise-free linear feedback control, which can effectively deal with the uncertainty of the manipulator system, suppress the external interference of the manipulator, and avoid control chattering. Ayeb et al. [32] designed an adaptive sliding mode controller based on an RBF neural network to improve the trajectory tracking performance of nonholonomic mobile robots and to avoid jitters. Al-Darraji et al. [33] designed an adaptive robust controller based on an RBF neural network, which takes into account high nonlinearity, high modeling errors, and the interference caused by payload and environmental conditions. It was able to combat effectively the nonlinear and uncertain problems of aerial robot arms. In [34], the adaptive control was used to update the parameters online in order to improve the asymptotic tracking performance of the uncertain nonlinear system, and the overall control process was introduced in detail; this description was drawn on here for the design of the controller.

3. Trajectory Planning Method

3.1. Problem Description

The design of the controllers has a significant effect on improving the trajectory tracking performance of robot manipulators. However, most of the controllers designed at this stage have been based on low degree of freedom manipulators, and the optimization of the tracking error of manipulators has also primarily been for low degree of freedom manipulators; further, and there have been error instabilities in certain trajectory optimization processes (seen in Figure 1 [28]).

Figure 1 presents the trajectory tracking error of the 3-DOF manipulator based on the RBF neural network controller. It can be seen from Figure 1 that only three manipulator joints were tracked; meanwhile, the trajectory error always exists during the trajectory operation, and does not diminish with time.

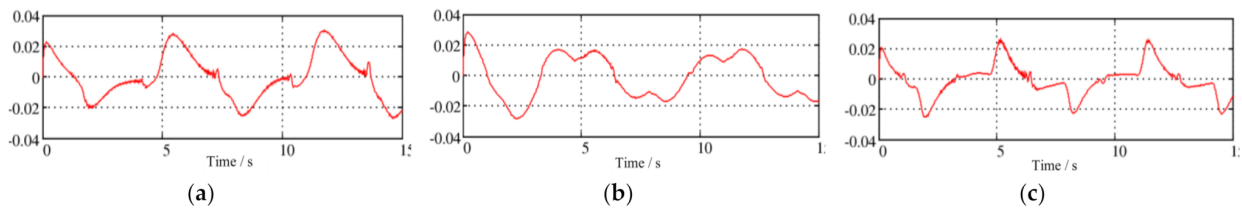


Figure 1. Trajectory tracking error of 3-DOF manipulator. (a) the tracking error of the expected trajectory and output trajectory of the first joint of the manipulator; (b) the tracking error of the expected trajectory and output trajectory of the second joint of the manipulator; (c) the tracking error of the expected trajectory and output trajectory of the third joint of the manipulator.

In addition, there has been little research on the design of multi-degree of freedom manipulator controllers. However, at this stage, the application scenarios of 6-DOF robots in various industries are increasing. Therefore, the trajectory optimization problem and the tracking error convergence problem of multi-degree of freedom manipulators need to be solved. It is imperative to design a controller that improves the trajectory tracking performance of 6-DOF manipulators.

3.2. Experiment Platform

The experimental setup for manipulator trajectory planning is shown in Figure 2. The upper computer and the control cabinet were connected through a network cable, and the control cabinet and the manipulator were connected through a power supply cable and a signal cable.

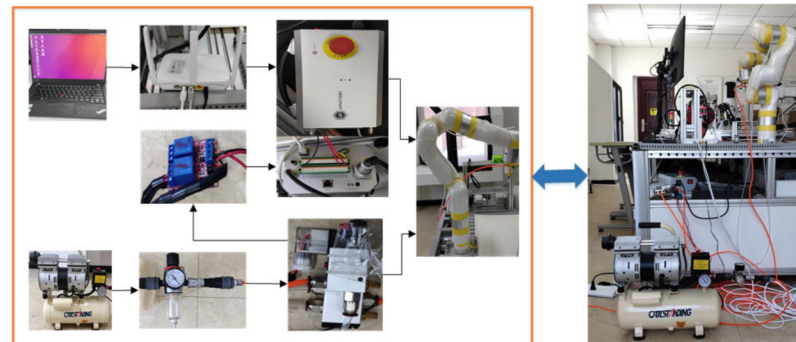


Figure 2. Experimental setup.

The main experimental platform for manipulator trajectory planning is shown in Figure 3, which mainly included a robot manipulator, control cabinet, upper computer and working platform.

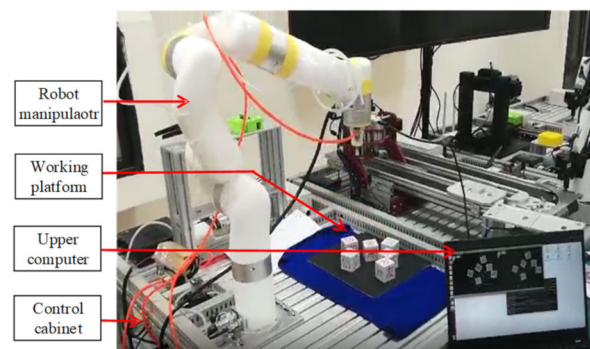


Figure 3. Experimental platform.

The main components of the robot control system were installed in the control cabinet. The layout of the control cabinet is shown in Figure 4, which mainly included a control

module, IO module, braking module, driver module, strong-current module and weak-current module, etc.

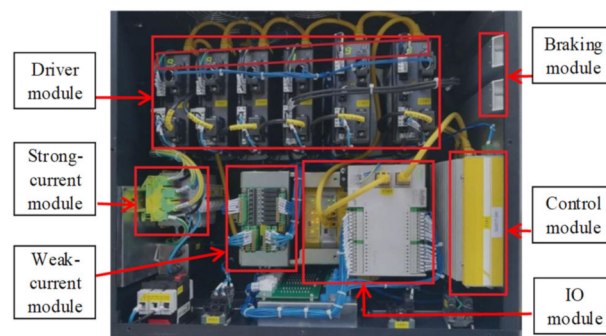


Figure 4. Control cabinet.

3.3. Trajectory Planning Architecture

The trajectory planning architecture of the robot manipulator is shown in Figure 5. This structure mainly consists of three parts.

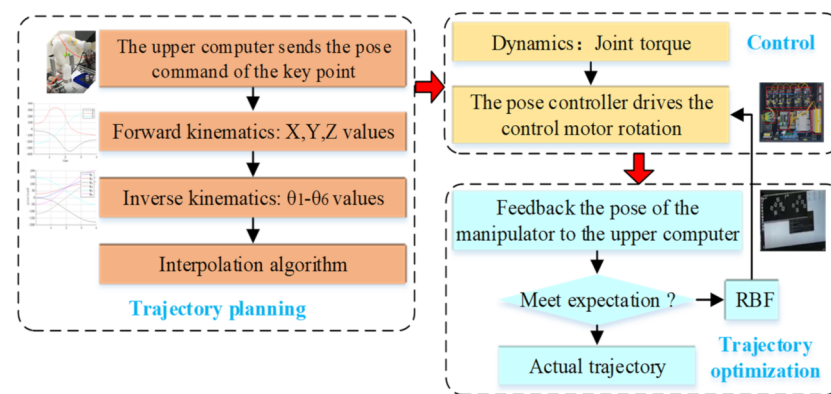


Figure 5. The trajectory planning structure.

The first stage is the trajectory planning stage. Firstly, the upper computer will send the pose commands of the key points of the robot manipulator. Secondly, the kinematics model of the manipulator is established based on the D–H method. Then, forward kinematics is used to find the x , y , and z values of the end effector, and inverse kinematics is used to find the θ_1 – θ_6 values of joint angle. Finally, the quintic polynomial interpolation algorithm is employed to obtain the ideal trajectory.

The second stage is the control system stage. Firstly, the dynamic model of manipulator is established based on Lagrange’s theorem. Secondly, the torque of each joint under the ideal trajectory is solved by dynamics. Then, the torque information is transmitted to the pose controller. Finally, the pose controller drives the control motor rotation to realize the movement of the manipulator.

The third stage is the trajectory optimization stage. Firstly, whether the pose of the end effector reaches the expected trajectory should be assessed: if it reaches the expected trajectory, the current trajectory should be taken as the actual trajectory; otherwise, the next step should be executed. Secondly, an adaptive robust controller based on an RBF neural network is designed, and the stability and convergence of the controller are analyzed based on a Lyapunov function. Then, the designed controller is used to optimize the tracking trajectory, and the new control command is transmitted to the pose controller. Finally, the trajectory for reaching the expected goal is taken as the actual trajectory of the manipulator.

3.4. Modeling

3.4.1. Kinematics Analysis

Position and angle analysis are the two main parts of kinematics modeling. Firstly, the structural parameters and link coordinate system of the manipulator are obtained based on the D–H method [35]. Secondly, the position of the end effector of the manipulator is obtained based on forward kinematics. Lastly, the angle of each joint of the manipulator is obtained based on inverse kinematics. Kinematics realizes the transformation of the manipulator's coordinates in Cartesian space and joint space, which lays the foundation for the trajectory planning of the manipulator.

The forward kinematics equation is derived by a homogeneous transformation matrix, which can be expressed as Equation (1):

$${}^0_6T = {}^0_1T(\theta_1){}_1^2T(\theta_2){}_2^3T(\theta_3){}_3^4T(\theta_4){}_4^5T(\theta_5){}_5^6T(\theta_6) = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

where p denote the position vector, a denotes the approach vector, o denotes the direction vector, and n denotes the normal vector.

The position can be expressed as Equation (2):

$${}^0_6P = {}^0_1T_1T_2T_3T_4T_5T_6TP \quad (2)$$

According to Equations (1)–(3) can then be obtained:

$$\begin{cases} n_x = c_1[c_{23}(c_4c_5c_6 - s_4s_6) - s_{23}s_5s_6] + s_1(s_4c_5c_6 + c_4s_6) \\ n_y = s_1[c_{23}(c_4c_5c_6 - s_4s_6) - s_{23}s_5s_6] - c_1(s_4c_5c_6 + c_4s_6) \\ n_z = -s_{23}(c_4c_5c_6 - s_4s_6) - c_{23}s_5s_6 \\ o_x = c_1[c_{23}(-c_4c_5c_6 - s_4s_6) + s_{23}s_5s_6] + s_1(-s_4c_5c_6 + c_4s_6) \\ o_y = s_1[c_{23}(-c_4c_5c_6 - s_4s_6) + s_{23}s_5s_6] - c_1(-s_4c_5c_6 + c_4s_6) \\ o_z = -s_{23}(-c_4c_5c_6 - s_4s_6) + c_{23}s_5s_6 \\ a_x = -c_1(c_{23}c_4c_5 + s_{23}c_5) - s_1s_4s_5 \\ a_y = -s_1(c_{23}c_4c_5 + s_{23}c_5) + c_1s_4s_5 \\ a_z = s_{23}c_4c_5 - c_{23}c_5 \\ p_x = c_1(a_1 + a_2c_2 + a_3c_{23} - d_4s_{23}) \\ p_y = s_1(a_1 + a_2c_2 + a_3c_{23} - d_4s_{23}) \\ p_z = d_1 - a_2s_2 - a_3s_{23} - d_4c_{23} \end{cases} \quad (3)$$

where the various parameters can be described as Equation (4):

$$\begin{cases} s_i = \sin(\theta_i) \\ c_i = \cos(\theta_i) \\ s_{ij} = \sin(\theta_i + \theta_j) = s_i c_j + c_i s_j \\ c_{ij} = \cos(\theta_i + \theta_j) = c_i c_j - s_i s_j \end{cases} \quad (4)$$

The inverse kinematics equation is obtained by the algebraic method. Each joint angle can be expressed as Equation (5):

$$\begin{cases} \theta_1 = A \tan 2(p_y, p_x) - A \tan 2(d_3, \pm \sqrt{p_x^2 + p_y^2 - d_3^2}) \\ \theta_2 = A \tan 2[-(a_3 + a_2c_3)p_z + (a_2s_3 - d_4)(c_1p_x + s_1p_y)] \\ \theta_3 = A \tan 2(a_3, d_4) - A \tan 2(K, \pm \sqrt{a_3^2 + d_4^2 - K^2}) \\ \theta_4 = A \tan 2(-a_x s_1 + a_y c_1, -a_x c_1 c_{23} - a_y s_1 c_{23} + a_z s_{23}) \\ \theta_5 = A \tan 2(s_5, c_5) \\ \theta_6 = A \tan 2(s_6, c_6) \end{cases} \quad (5)$$

3.4.2. Dynamics Analysis

Dynamics analysis forms the basis of the manipulator's controller. To date, many methods regarding the dynamics analysis of manipulators have been developed. The common methods include the Newton Euler equation, Lagrange's equation, Kane's equation, Appel's equation, Routh's equation, and so on [36]. The dynamic model of the mechanical system is derived by Lagrange's theorem and can be described as Equation (6):

$$M(q)\ddot{q} + V(q, \dot{q})\dot{q} + G(q) + F(\dot{q}) + d_s = \tau + \tau_e \quad (6)$$

where q is the joint angular displacement vector, \dot{q} is the joint angular velocity vector, \ddot{q} is the joint angular acceleration vector, $M(q)$ is the 6×6 order positive definite inertia matrix, $V(q, \dot{q})$ is the 6×6 order inertia matrix, $G(q)$ is the 6×1 order gravity matrix, $F(\dot{q})$ is the 6×1 order friction matrix, d_s is the external interference, τ_e is the measurable environmental torque exerted on the robot manipulators, and τ is the control input.

Suppose that the dynamic model of the robot manipulator has unknown but bounded parameters and modeling errors; then, the robot dynamics part and the measurable environmental torque described using the RBF neural network structure can be written as follows:

$$M(q)\ddot{q} + V(q, \dot{q})\dot{q} + G(q) + F(\dot{q}) = W^T H(q, \dot{q}, \ddot{q}, t) \quad (7)$$

where W are the unknown parameters for robot manipulators, and H is the RBF neural network matrix.

The measurable environmental torque described using the RBF neural network structure can be written as follows:

$$\tau_e = W_e^T H_e(x_e) = W_e^T H_e(q, \dot{q}, \ddot{q}) \quad (8)$$

where W_e are the unknown parameters for robot manipulators, and H_e is the RBF neural network matrix of the environment. This model has the general characteristics of a RBF neural network, and can describe different actual environments, including the free motion condition when $W_e = 0$.

The optimal estimation parameter for estimating W_e is defined as follows:

$$\widehat{W}_e = \arg \min_{W_e \in \lambda_{e0}} \left[\sup_{x_e \in \lambda_e} \left| W_e^T H_e(x_e) - \widehat{W}_e^T H_e(x_e) \right| \right] \quad (9)$$

where λ_{e0} is the bounded set of W_e , λ_e is the bounded set of x_e , and \widehat{W}_e is updated online by Equation (10) to guarantee an acceptable estimation of W_e .

$$\dot{\widehat{W}}_e = K_e F_e \widetilde{W}_e \quad (10)$$

where $K_e > 0$, $F_e > 0$, \widetilde{W}_e is the environmental parameters estimation error, which can be described as follows:

$$\widetilde{W}_e = W_e - \widehat{W}_e \quad (11)$$

We set x_e and τ_e as the input and output of the RBF neural network respectively. Then, the optimal estimation parameters \widehat{W}_e can be obtained. In this way, we can use the non-power environmental parameters \widehat{W}_e in the controller to replace the traditional environmental torque τ_e , thereby avoiding the problem of passivity.

3.5. Trajectory Planning Algorithm

The trajectory planning algorithm used for the robot manipulator is a quintic polynomial interpolation algorithm [37], which can be described as Equation (12):

$$\begin{cases} \theta(t) = a_0 + a_1t + a_2t^2 + a_3t^3 + a_4t^4 + a_5t^5 \\ \dot{\theta}(t) = a_1 + 2a_2t + 3a_3t^2 + 4a_4t^3 + 5a_5t^4 \\ \ddot{\theta}(t) = 2a_2 + 6a_3t + 12a_4t^2 + 20a_5t^3 \end{cases} \quad (12)$$

where t denotes the time, $\theta(t)$ denotes the Angular displacement, $\dot{\theta}(t)$ denotes the angular velocity, and $\ddot{\theta}(t)$ denotes the angular acceleration.

The constraint condition of each coefficient of the quintic polynomial interpolation algorithm is described as Equation (13):

$$\begin{cases} \theta(t_0) = \theta_0 = a_0 \\ \theta(t_f) = \theta_f = a_0 + a_1t_f + a_2t_f^2 + a_3t_f^3 + a_4t_f^4 + a_5t_f^5 \\ \dot{\theta}(t_0) = \dot{\theta}_0 = a_1 \\ \dot{\theta}(t_f) = \dot{\theta}_f = a_1 + 2a_2t_f + 3a_3t_f^2 + 4a_4t_f^3 + 5a_5t_f^4 \\ \ddot{\theta}(t_0) = \ddot{\theta}_0 = 2a_2 \\ \ddot{\theta}(t_f) = \ddot{\theta}_f = 2a_2 + 6a_3t_f + 12a_4t_f^2 + 20a_5t_f^3 \end{cases} \quad (13)$$

When the constraint condition is satisfied, that is, when (13) is substituted into (12), Equation (14) can then be obtained:

$$\begin{cases} a_0 = \theta_0 \\ a_1 = \dot{\theta}_0 \\ a_2 = \frac{\ddot{\theta}_0}{2} \\ a_3 = \frac{1}{2t_f^3} [20\theta_f - 20\theta_0 - (8\dot{\theta}_f + 12\dot{\theta}_0)t_f - (3\ddot{\theta}_0 - \ddot{\theta}_f)t_f^2] \\ a_4 = \frac{1}{2t_f^4} [30\theta_f - 30\theta_0 + (14\dot{\theta}_f + 16\dot{\theta}_0)t_f - (3\ddot{\theta}_0 - 2\ddot{\theta}_f)t_f^2] \\ a_5 = \frac{1}{2t_f^5} [12\theta_f - 12\theta_0 - (6\dot{\theta}_f + 6\dot{\theta}_0)t_f - (\ddot{\theta}_0 - \ddot{\theta}_f)t_f^2] \end{cases} \quad (14)$$

4. RBF Neural Network

4.1. RBF Neural Network Architecture

An RBF network [38] is a three-layer feedforward neural network with a radial basis function as its activation function; its structure is shown in Figure 6. It has been proven that the errors of arbitrary continuous functions can be reduced by RBF neural networks: that is, their nonlinear function approximation ability is strong. They can greatly speed up the learning rate and avoid local minima; they also have higher response speeds that are 1000 to 10,000 times faster than BP neural networks.

The three-layered RBF neural network consists of the input layer, hidden layer, and output layer. The input layer is composed of signal source nodes and transmits input excitation to the hidden layer; the hidden layer adopts gauss radial basis functions to map the low-dimensional input to the high-dimensional space and performs curve fitting; the output layer adopts a linear transformation function to perform weighted evaluation on the hidden layer signal in order to obtain the output value.

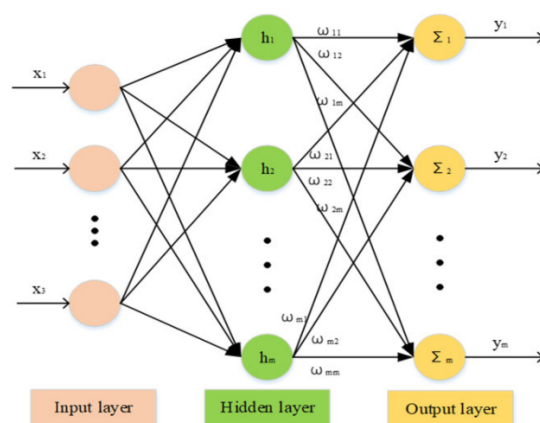


Figure 6. Structure of RBF neural network.

In the RBF network structure, the following notations are used [39]:

$X = [x_1, x_2, \dots, x_n]^T$ is the input vector in the input layer, $W = [w_1, w_2, \dots, w_m]^T$ is the weight vector, $H = [h_1, h_2, \dots, h_m]^T$ is the radial basis vector, and h_j is the Gaussian basis function, which can be calculated as follows:

$$h_j = \exp\left(-\frac{\|X - c_j\|^2}{2\sigma_j^2}\right) \quad j = 1, 2, \dots, m \tag{15}$$

where $c_j = [c_1, c_2, \dots, c_m]$ is the central vector of the j -th node in the network, and σ_j is the mean deviation of the j -th node in the network. According to the structure chart, the input vector of the RBF neural network is $X = [x_1, x_2, \dots, x_n]^T$. The output of the RBF network can be calculated as follows:

$$y(t) = \sum_{i=1}^m w_i h_i \tag{16}$$

4.2. Adaptive Robust Controller Design Based on RBF Neural Network

In robot control, more and more researchers are designing new controllers for nonlinear problems. Because of the problems of control chattering and external interference in multi-degree of freedom manipulators, a stable controller needs to be designed.

In this study, we designed an adaptive robust controller τ based on an RBF neural network, which can achieve good tracking performance under nonlinearity and uncertainty. As indicated in the dynamic model of the system in Equation (6), the manipulator’s trajectory tracking aims to make the joint angle vector $q(t) = [q_1(t), q_2(t), \dots, q_n(t)]$ track the designated joint angle vector $q_d(t) = [q_{d1}(t), q_{d2}(t), \dots, q_{dn}(t)]$.

The trajectory tracking error and error function are defined as Equations (17) and (18), respectively.

$$e(t) = q_d(t) - q(t) \tag{17}$$

$$r = \dot{e} + \Lambda e \tag{18}$$

where r, Λ is the positive diagonal matrix.

Substituting (17) and (18) into (6), Equations (19) and (20) are obtained:

$$\dot{q} = -r + \dot{q}_d + \Lambda e \tag{19}$$

$$\begin{aligned} M\dot{r} &= M(\dot{q}_d - \dot{q} + \Lambda e) = M(\dot{q}_d + \Lambda e) - M\dot{q} = M(\dot{q}_d + \Lambda e) + V\dot{q} + G + F + d_s - \tau - \tau_e \\ &= M(\dot{q}_d + \Lambda e) - Vr + V(\dot{q}_d + \Lambda e) + G + F + d_s - \tau - \tau_e \\ &= -Vr - \tau - \tau_e + f + d_s \end{aligned} \tag{20}$$

where f include dynamics parameters and are usually unknown in the actual system, and they can be expressed as follows:

$$f(x) = M(\ddot{q}_d + \wedge \dot{e}) + V(q_d + \wedge e) + G + F \quad (21)$$

It can be seen from the above equations that an accurate mathematical model for manipulators is very necessary, but this is difficult to obtain, for manipulators are non-linear and uncertain systems. Hence, there will be a great error in the calculated result of $f(x)$. To solve this problem, an RBF neural network is used to approximate $f(x)$. Suppose the input of the RBF neural network is described as follows:

$$x = \left[e^T, \dot{e}^T, q_d^T, \dot{q}_d^T, \ddot{q}_d^T \right] \quad (22)$$

The ideal output of the RBF neural network is as follows:

$$h_j = \exp\left(-\frac{\|X - c_j\|^2}{2\sigma_j^2}\right) \quad (23)$$

$$f(x) = W^T H(x) + \varepsilon(x) \quad (24)$$

where W is the weight matrix of the ideal neural network, and $\varepsilon(x) = [\varepsilon_1(x), \varepsilon_2(x), \dots, \varepsilon_n(x)]^T$ is the approximation error of the ideal neural network.

Suppose the actual output of the RBF neural network is:

$$\hat{f}(x) = \hat{W}^T H(x) \quad (25)$$

Given $\tilde{W} = W - \hat{W}$, \hat{W} is the weight for the actual approximation, \tilde{W} is the error between the ideal weight and the actual weight, and $\hat{f}(x)$ is the actual approximation value of the neural network to $f(x)$.

Then, the adaptive robust controller τ based on the RBF neural network can be designed as Equation (26):

$$\tau = \hat{W}^T H(x) + K_v r - v - \tau_e \quad (26)$$

where $K_v > 0$, and τ_e is the measurable environmental torque.

Meanwhile, the adaptive law for online and real-time estimation of the parameters of the radial basis function neural network can be designed as Equation (27):

$$\dot{\hat{W}} = FH(x)r^T - KF\|r\|\hat{W} \quad (27)$$

where $K > 0$, and $F > 0$.

According to the above expression, Equation (28) can be obtained:

$$M\dot{r} = -(K_v + V)r + \zeta_1 \quad (28)$$

where $\zeta_1 = \tilde{W}^T H(x) + (\varepsilon + d_s) + v$, v is the robust term used to cope with the approximation error of the RBF neural network and the system external disturbance and modeling error. v is designed as Equation (29):

$$v = -(\varepsilon_n + b_d)\text{sgn}(r) \quad (29)$$

where b_d is the upper bound of the interference error, and ε_n is the upper bound of the approximation error: $\|\varepsilon\| \leq \varepsilon_n$, $d_s \leq b_d$.

In order for the robot manipulator to achieve good control performance while also ensuring stability, combined with the adaptive law and the control law with robust terms, the lower bound of K_v must conform to the Equation (30):

$$K_{v\min} \geq \frac{KW_{\max}^2}{4\|r\|} \quad (30)$$

Then, the position signal, velocity signal, and acceleration signal are all bounded, and after this, the robot manipulator system tends to be stable. As time increases, the tracking error gradually tends to zero: that is, $e(t) \rightarrow 0$ as $t \rightarrow \infty$.

4.3. Stability and Convergence Analysis

For the adaptive robust control of the robot manipulator based on the RBF neural network, the Lyapunov function [40] is defined as Equation (31):

$$L = \frac{1}{2}r^TMr + \frac{1}{2}\text{tr}(\tilde{W}^TF^{-1}\tilde{W}) \quad (31)$$

Then, the derivative of L can be derived as Equation (32):

$$\dot{L} = r^T\dot{M}r + \frac{1}{2}r^T\dot{M}r + \text{tr}(\tilde{W}^T\dot{F}^{-1}\tilde{W}) \quad (32)$$

Substituting (28) into (31), Equation (33) can be obtained:

$$\dot{L} = -r^TK_vr + \frac{1}{2}r^T(\dot{M} - 2V)r + \text{tr}\tilde{W}^T(F^{-1}\dot{\tilde{W}} + Hr^T) + r^T(\varepsilon + d_s + v) \quad (33)$$

Substituting the adaptive law (27) into (33), we can then obtain Equation (34):

$$\begin{aligned} \dot{L} &= -r^TK_vr + r^T(\varepsilon + d_s + v) + \text{tr}\tilde{W}^T(-F^{-1}F Hr^T + KF^{-1}F\|r\|\tilde{W} + Hr^T) \\ &= -r^TK_vr + r^T(\varepsilon + d_s + v) + K\|r\|\text{tr}\tilde{W}(W - \tilde{W}) \end{aligned} \quad (34)$$

where the trace of matrix $\tilde{W}(W - \tilde{W})$ can be described as Equation (35):

$$\text{tr}\tilde{W}^T(W - \tilde{W}) = (\tilde{W}, W) - \|\tilde{W}\|^2 \leq \|\tilde{W}\|\|W\| - \|\tilde{W}\|^2 \quad (35)$$

Then, the relationship of \dot{L} can be derived as Equation (36):

$$\begin{aligned} \dot{L} &= -r^TK_vr + r^T(\varepsilon + d_s + v) + K\|r\|\text{tr}\tilde{W}(W - \tilde{W}) \\ &\leq -K_{v\min}\|r\|^2 + r^T(\varepsilon + d_s + v) + K\|r\|\|\tilde{W}\|(W_{\max} - \|\tilde{W}\|) \\ &\leq -K_{v\min}\|r\|^2 + r^T(\varepsilon + d_s) - \|r\|(\varepsilon_n + b_d) + K\|r\|\|\tilde{W}\|(W_{\max} - \|\tilde{W}\|) \\ &\leq -\|r\|\left[K_{v\min}\|r\| + K\|\tilde{W}\|(\|\tilde{W}\| - W_{\max})\right] \\ &= -\|r\|\left[K(\|\tilde{W}\| - \frac{W_{\max}}{2}) - \frac{KW_{\max}^2}{4} + K_{v\min}\|r\|\right] \end{aligned} \quad (36)$$

When the lower bound of K_v meets the Equation (30), Equation (37) can then be obtained:

$$\dot{L} \leq -\|r\|\left[K(\|\tilde{W}\| - \frac{W_{\max}}{2}) - \frac{KW_{\max}^2}{4} + K_{v\min}\|r\|\right] \leq 0 \quad (37)$$

As $L \geq 0$, $\dot{L} \leq 0$; therefore, L is positive and bounded. Meanwhile, $M(q)$ is also positive and bounded, which indicates that $r(t)$, \tilde{W} , and \hat{W} are also bounded. Moreover, when the value of the Lyapunov function is equal to 0, the value of the error function

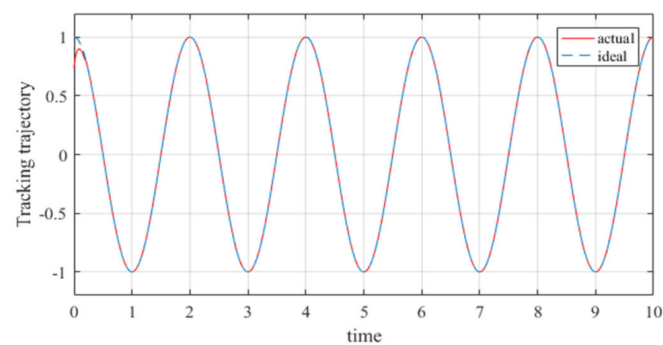
is also equal to 0: that is, $r = 0$ when $\dot{L} = 0$. According to Barbalat's lemma, the robot manipulator system is asymptotically stable. Therefore, as time increases, the tracking error, the derivative of the tracking error, and the error function all also gradually tend to zero: that is, $e(t) \rightarrow 0$, $\dot{e}(t) \rightarrow 0$, and $r \rightarrow 0$ as $t \rightarrow \infty$.

5. Simulation and Experiment Results

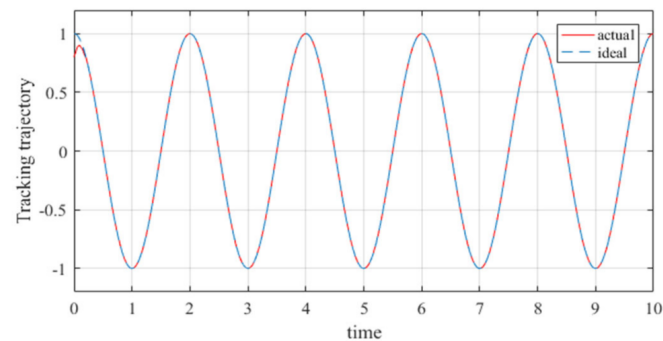
5.1. Trajectory Tracking Simulation

5.1.1. Simulation Results

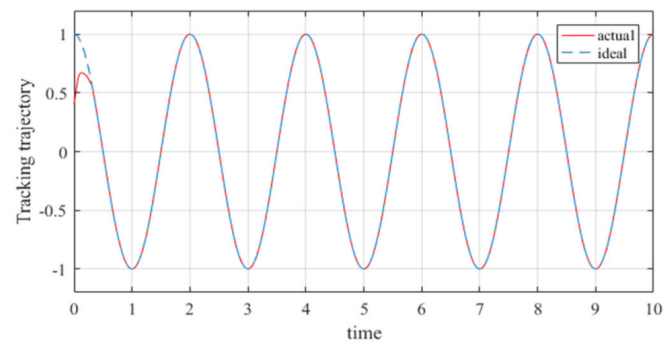
The trajectory tracking simulation results are shown in the figures below. Figure 7 shows the trajectory of the joint angle tracking of the adaptive robust controller based on the RBF neural network. Figure 8 shows the trajectory of the joint position tracking of the adaptive robust controller based on the RBF neural network. The red line represents the actual trajectory, and the blue dotted line represents the ideal trajectory.



(a)

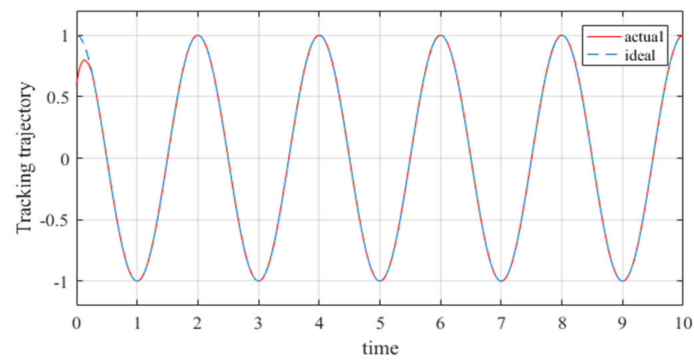


(b)

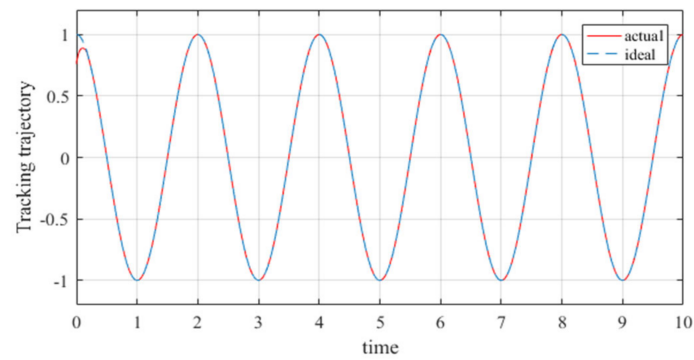


(c)

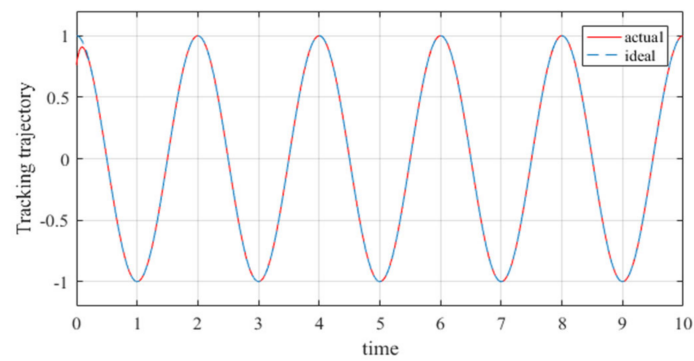
Figure 7. Cont.



(d)

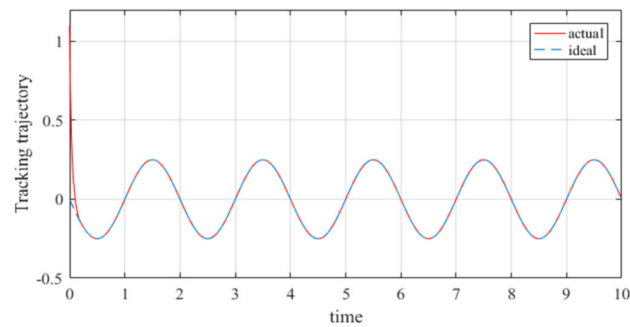


(e)



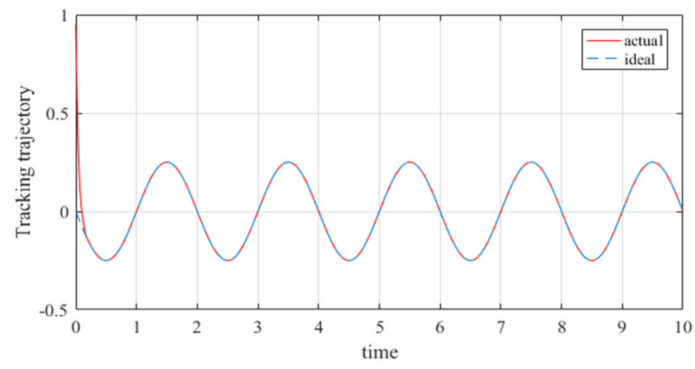
(f)

Figure 7. Joint angle tracking of the proposed controller: (a) angle tracking of joint 1; (b) angle tracking of joint 2; (c) angle tracking of joint 3; (d) angle tracking of joint 4; (e) angle tracking of joint 5; (f) angle tracking of joint 6.

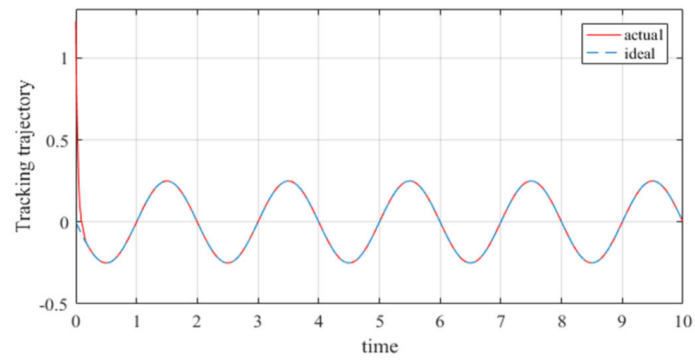


(a)

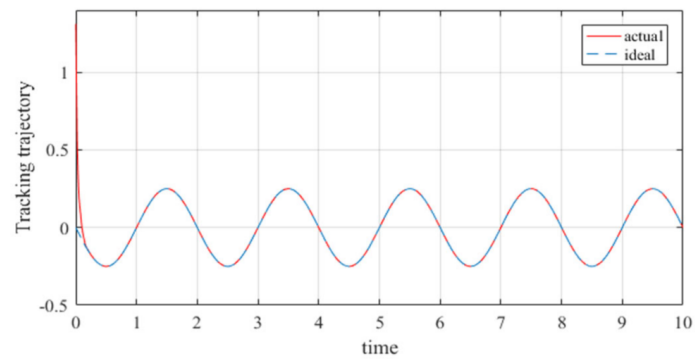
Figure 8. Cont.



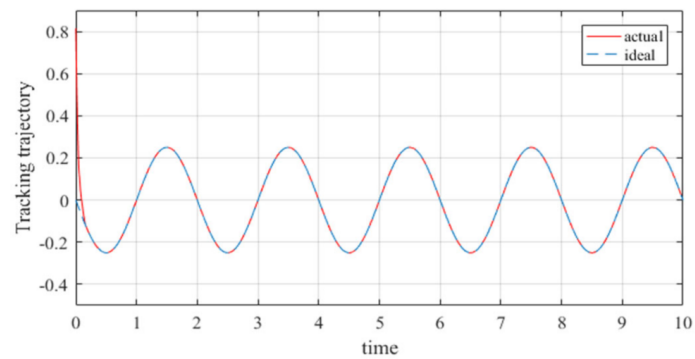
(b)



(c)



(d)



(e)

Figure 8. Cont.

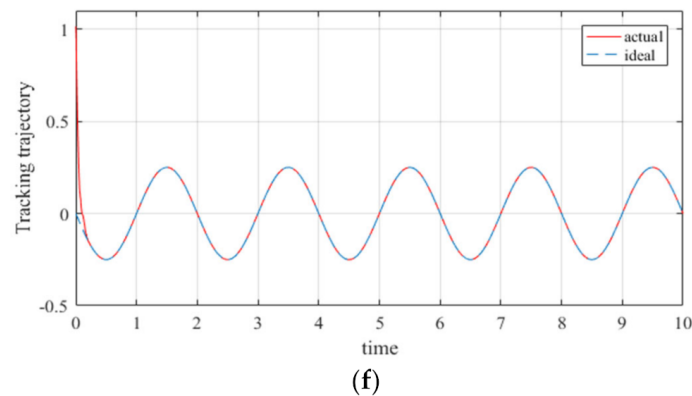


Figure 8. Joint position tracking of the proposed controller: (a) position tracking of joint 1; (b) position tracking of joint 2; (c) position tracking of joint 3; (d) position tracking of joint 4; (e) position tracking of joint 5; (f) position tracking of joint 6.

5.1.2. Simulation Analysis

Based on the simulation results, the following conclusions can be reached.

- (1) In Figures 7 and 8, the estimation $f(x)$ with the RBF neural network is expressed by $\hat{f}(x)$. We can see that $\hat{f}(x)$ almost approximated to $f(x)$ after 0.3 s, and the error was in an acceptable range.
- (2) It can be seen from Figures 7 and 8 that the adaptive robust controller based on the RBF neural network tracked the trajectory of the manipulator, and therefore the proposed controller improved the response time while reducing the adjustment time.
- (3) The tracking errors in the simulation strictly converged to zero; this means that the proposed controller can guarantee the stability of manipulators in real applications. In other words, the simulation results reveal that the proposed controller is effective for multi-degree of freedom manipulators faced with uncertainties and external disturbances.

Meanwhile, the controller proposed in this study was compared with other controllers, and the comparison results are shown in Table 1.

Table 1. Controller comparison results.

Controller	Manipulator	Maximum Tracking Error	Error Approach Time	Joint Position Tracking	Joint Angle Tracking	Environmental Interference
Robust controller [28]	3 DOF	0.028 rad	Near 1 s	No	Yes	Not considered
Adaptive controller [29]	2 DOF	Near 0.4 rad	Near 6 s	No	Yes	Not considered
Sliding mode controller [30]	2 DOF	Near 0.7 rad	Near 2.5 s	Yes	No	Not considered
Adaptive sliding controller [41]	2 DOF	Near 0.43 rad	Near 0.6 s	No	Yes	Considered
Proposed controller	6 DOF	Near 0.5 rad	Near 0.2 s	Yes	Yes	Considered

From Table 1, it can be seen that the controller designed in this study not only considers joint angle tracking and joint position tracking in the trajectory tracking of 6-DOF manipulators, but also considers the external environment interference to ensure the credibility of their trajectories. Compared with other controllers, the trajectory tracking error of this controller is not the smallest, but the error approximation time is very short, and can be applied to a 6-DOF manipulator.

5.1.3. Simulation Discussion

In the process of manipulator trajectory tracking, there was only a large error in the initial state, and the error decreased rapidly in a very short time. Then, the trajectory tracking error gradually converged to 0, and there was no sudden change in the error. This indicates that the adaptive robust controller based on the RBF neural network designed in this study achieved good results in realizing the trajectory tracking of the 6-DOF manipulator, has extremely high stability, and improves the trajectory tracking performance of the manipulator.

5.2. Trajectory Planning Simulation

5.2.1. Simulation Results

It has been proven that the RBF neural network can fit discrete points with minimal errors, so it can therefore be applied to manipulator trajectory planning. We adopted the quintic polynomial interpolation algorithm to plan the trajectory of the robot manipulator.

The trajectory planning simulation results are shown in the figures below. Figure 9 shows the angular displacement trajectory of each joint of the robot manipulator. Figure 10 shows the angular velocity trajectory of each joint of the robot manipulator. Figure 11 shows the angular acceleration trajectory of each joint of the robot manipulator.

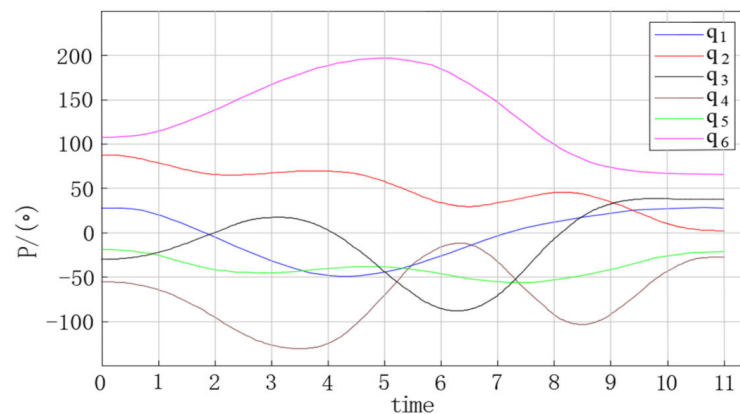


Figure 9. Angular displacement curve of each joint of the manipulator.

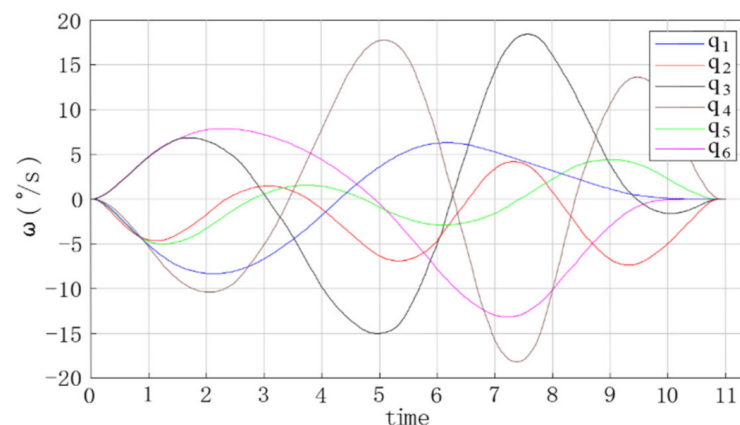


Figure 10. Angular velocity curve of each joint of the manipulator.

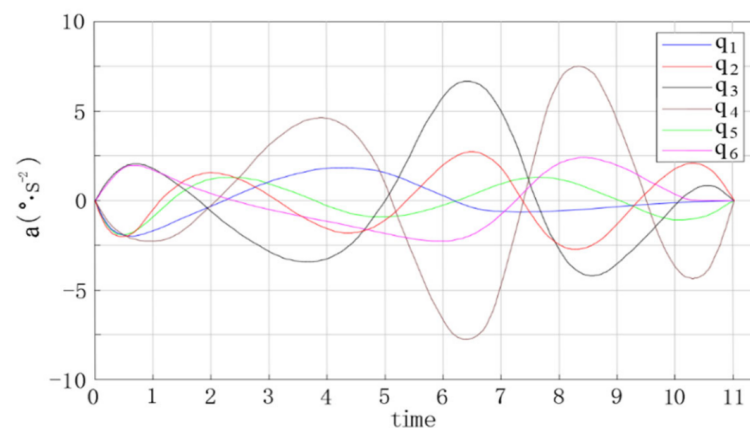


Figure 11. Angular acceleration curve of each joint of the manipulator.

5.2.2. Simulation Analysis

Based on the trajectory planning simulation results, the following conclusions can be reached.

- (1) In Figures 9–11, the trajectories of the joint angular displacement, angular velocity, and angular acceleration are smooth, and there are no jumps. This indicates that there were no jitter or impact problems in the trajectory planning of the multi-degree of freedom manipulator.
- (2) It can be seen from Figures 10 and 11 that the angular velocity and angular acceleration of each joint of the manipulator at the starting and ending position were all 0, which indicates that the manipulator can run smoothly when starting and stopping movement, as well as during the entire process of completing work tasks.
- (3) It can be seen from Figures 10 and 11 that the velocity and acceleration of joint 3 and joint 4 of the 6-DOF manipulator varied drastically from 3 to 9 s, indicating that the method proposed in this study can improve the trajectory planning speed and shorten the trajectory planning time.

5.2.3. Simulation Discussion

During the trajectory planning of the 6-DOF robot manipulator, the positions of the six joints were constantly changing, which indicates that the method proposed in this paper can realize the full scheduling of the 6-DOF robot manipulator. The velocity and acceleration values of the initial state and the end state of the trajectory planning were all 0, which represents the smooth end of a trajectory planning experiment. The position, velocity, and acceleration curves of the trajectory planning were smooth, indicating that the method proposed in this paper can effectively improve the stability, speed, and accuracy of trajectory planning.

5.3. Trajectory Planning Experiment

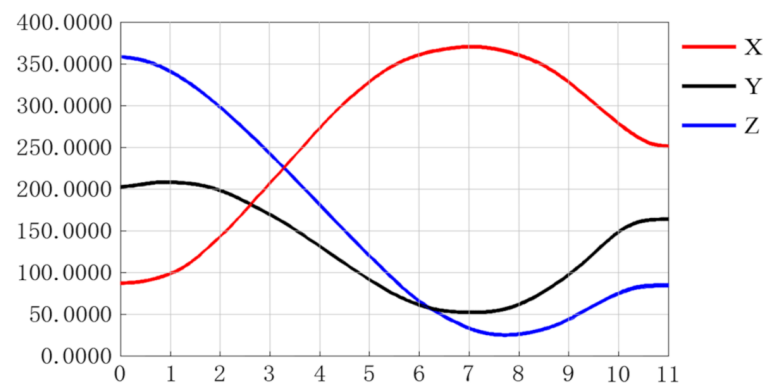
5.3.1. Experiment Results

The trajectory planning experiment data of the manipulator passing through 12 space nodes are shown in Table 2, which shows the actual variables of each joint and the coordinates of the end effector under the 12 space nodes of the manipulator.

According to the Cartesian space coordinates in Table 2, the x -axis, y -axis, and z -axis displacement curves of the end effector of the manipulator are drawn using the quintic polynomial interpolation function, as shown in Figure 12.

Table 2. Manipulator motion data.

Position	Actual Joint Variables (°)						End Effector Coordinates (mm)		
	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	X	Y	Z
0	32.0364	84.3602	33.0139	−52.6213	−21.3309	103.4506	78.2096	202.0341	357.6408
1	21.4961	80.1347	23.8134	−61.2139	−25.6102	109.1382	99.8657	208.3627	339.5047
2	−3.6823	67.2143	0.9148	−96.0141	−45.8168	142.8134	146.3812	199.0349	298.7648
3	−34.3147	68.8139	20.6127	−124.4116	−47.2314	168.5148	207.9973	171.3106	243.3942
4	−48.3015	70.5126	1.8631	−123.1671	−44.1861	183.7152	273.6841	132.6172	179.3016
5	−46.1137	61.8217	−46.2916	−71.1029	−43.2319	192.3185	332.4615	89.3364	122.3657
6	−23.6124	42.8133	−86.1991	−11.7163	−48.3161	180.7162	362.5973	62.8249	65.8143
7	−0.8135	42.2019	−71.8634	−31.9013	−52.3172	148.6173	368.7526	52.6815	36.4265
8	16.2643	47.9103	−9.8638	−47.3129	−51.0192	109.8167	363.0214	63.8462	28.8318
9	23.6245	43.1081	42.4813	−47.6013	−44.5148	73.9216	330.8106	98.1835	48.2154
10	28.3012	13.2164	43.8156	−48.1176	−24.6137	19.2131	278.6105	149.0263	77.3167
11	29.6148	1.5018	43.8156	−16.9135	−23.1772	13.2131	251.0648	166.8019	83.3182

**Figure 12.** End effector displacement curve.

Meanwhile, the overall process of the manipulator trajectory planning experiment is shown in Figure 13.

5.3.2. Experiment Analysis

Based on the trajectory planning simulation results, the following conclusions can be reached.

- (1) It can be seen from Figure 12 that the three coordinate changes of the end effector were smooth curves, which indicates the rationality of the forward and inverse kinematics model designed in this paper.
- (2) Figure 13a represents the initial state of manipulator trajectory planning, Figure 13b represents the state when picking up the object, Figure 13c,d represents the state of the moving trajectory, Figure 13e represents the state when the object is placed, and Figure 13f represents the end state of the trajectory.
- (3) It can be seen from Figure 13 that the manipulator ran smoothly, and the trajectory was smooth and continuous in the wood block grasping experiment, which verifies the feasibility of the method proposed in this paper.

5.3.3. Experiment Discussion

The trajectory planning experiment for the 6-DOF manipulator was conducted to verify the feasibility of the method proposed in this paper. In the actual experiment, it ran smoothly without sudden changes in speed, and could grasp the target, which indicates that the method proposed in this paper not only has theoretical significance, but also has practical application value. It can make up for part of the current gap in the manipulator research field and can provide corresponding technical theoretical support for multi-degree of freedom manipulator trajectory planning.

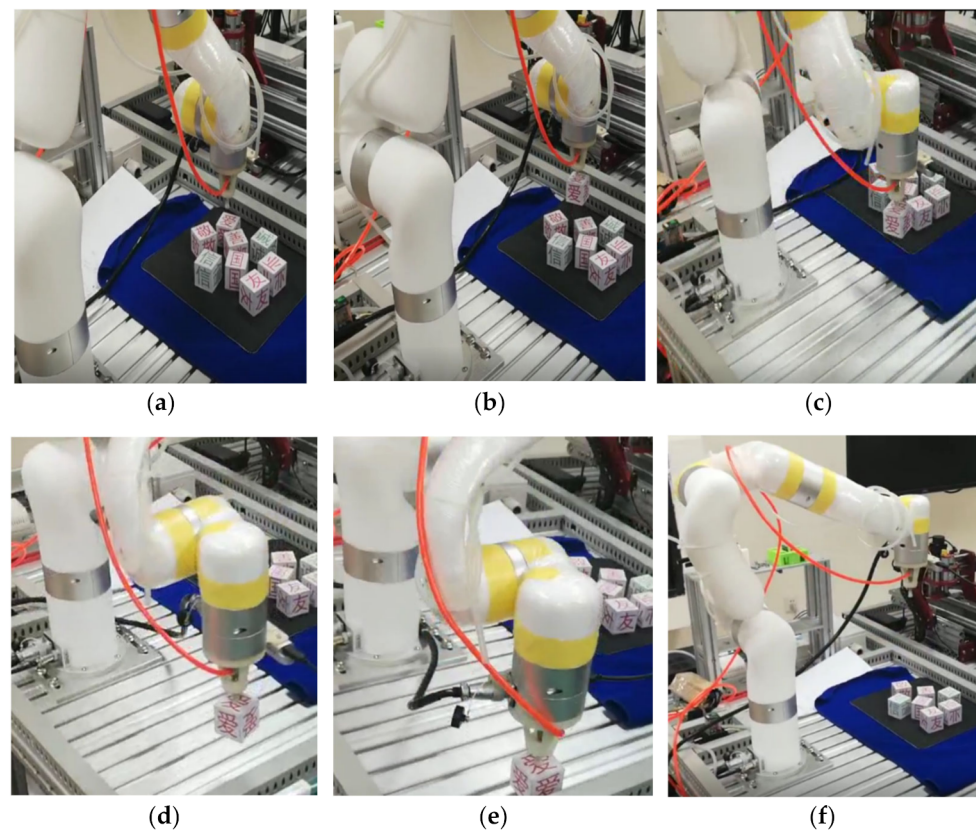


Figure 13. Trajectory planning experiment of the manipulator: (a–f) represent different states of the manipulator in the process of trajectory planning.

6. Conclusions

In this paper, we proposed a powerful trajectory planning method for robot manipulators, which is based on an RBF neural network. The proposed method was evaluated by two simulations. The first simulation evaluated the precision of the trajectory tracking of the manipulator, and the second simulation evaluated the motion stability of the trajectory planning of the manipulator. In addition, the proposed method was verified by an experiment. The experiment not only verified the rationality of the kinematics and dynamics model, but also verified the feasibility and effectiveness of the proposed method. The simulation and experiment results proved that the proposed method can improve the trajectory tracking accuracy and motion efficiency of the manipulator. Meanwhile, the designed controller is robust, able to withstand not only external disturbances but also parameter uncertainties. This paper focuses on the trajectory planning of multi-degree of freedom manipulators and makes corresponding explorations into the development of robots.

In the future, further tests are essential for performance evaluation of the proposed control approach. We will take the motion time and energy consumption into account to obtain a comprehensive optimal trajectory. Meanwhile, we will also continue to study and optimize the control strategies of manipulators.

Author Contributions: Conceptualization, S.L. and Q.S.; methodology, Q.S., A.Z. and L.Z.; software, Q.S. and J.Y.; validation, Q.S., Q.B. and L.Z.; formal analysis, Q.S.; investigation, Q.S. and X.Z.; resources, Q.S.; data curation, Q.S. and X.Z.; writing—original draft preparation, Q.S.; writing—review and editing, Q.S., S.L. and J.Y.; visualization, Q.S.; supervision, S.L.; project administration, S.L.; funding acquisition, S.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially funded by the National Key R&D Program of China (No. 2020YFB1713300, No. 2018AAA0101803), the Higher Education Project of Guizhou Province (No. [2020]005, No. [2020]009), and the Science and Technology Project of Guizhou Province (No. [2019]3003).

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. Lefebvre, T.; Crevecoeur, G. On entropy regularized path integral control for trajectory optimization. *Entropy* **2020**, *22*, 1120. [[CrossRef](#)]
2. Rybus, T. Obstacle avoidance in space robotics: Review of major challenges and proposed solutions. *Prog. Aerosp. Sci.* **2018**, *101*, 31–48. [[CrossRef](#)]
3. Omisore, O.M.; Han, S.; Al-Handarish, Y.; Du, W.; Duan, W.; Akinyemi, T.O.; Wang, L. Motion and trajectory constraints control modeling for flexible surgical robotic systems. *Micromachines* **2020**, *11*, 386. [[CrossRef](#)] [[PubMed](#)]
4. Bai, Q.; Li, S.; Yang, J.; Song, Q.; Li, Z.; Zhang, X. Object detection recognition and robot grasping based on machine learning: A survey. *IEEE Access* **2020**, *8*, 181855–181879. [[CrossRef](#)]
5. Chembuly, V.V.M.J.S.; Voruganti, H.K. Trajectory Planning of Redundant Manipulators Moving along Constrained Path and Avoiding Obstacles. *Procedia Comput. Sci.* **2018**, *133*, 627–634. [[CrossRef](#)]
6. Liu, A.; Zhao, H.; Song, T.; Liu, Z.; Wang, H.; Sun, D. Adaptive control of manipulator based on neural network. *Neural Comput. Appl.* **2021**, *33*, 4077–4085. [[CrossRef](#)]
7. Dong, S.; Chen, G.; Liu, M.; Wu, Z.G. Robust adaptive H_∞ control for networked uncertain semi-Markov jump nonlinear systems with input quantization. *Sci. China Inf. Sci.* **2022**, *65*, 1–2. [[CrossRef](#)]
8. Zhang, N.; Canini, K.; Silva, S.; Gupta, M. Fast Linear Interpolation. *ACM J. Emerg. Technol. Comput. Syst.* **2021**, *17*, 2. [[CrossRef](#)]
9. Dinçer, Ü.; Çevik, M. Improved trajectory planning of an industrial parallel mechanism by a composite polynomial consisting of Bézier curves and cubic polynomials. *Mech. Mach. Theory* **2019**, *132*, 248–263. [[CrossRef](#)]
10. Kumar Kashyap, A.; Parhi, D.R. Multi-objective trajectory planning of humanoid robot using hybrid controller for multi-target problem in complex terrain. *Expert Syst. Appl.* **2021**, *179*, 1–22. [[CrossRef](#)]
11. Zhang, T.; Zhang, M.; Zou, Y. Time-optimal and Smooth Trajectory Planning for Robot Manipulators. *Int. J. Control. Autom. Syst.* **2021**, *19*, 521–531. [[CrossRef](#)]
12. Kim, H.; Kim, B.K. Energy-optimal transport trajectory planning and online trajectory modification for holonomic robots. *Asian J. Control* **2020**, 1–16. [[CrossRef](#)]
13. Chai, R.; Tsourdos, A.; Savvaris, A.; Chai, S.; Xia, Y.; Chen, C.L.P. Six-DOF Spacecraft Optimal Trajectory Planning and Real-Time Attitude Control: A Deep Neural Network-Based Approach. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *31*, 5005–5013. [[CrossRef](#)] [[PubMed](#)]
14. Huang, J.; Hu, P.; Wu, K.; Zeng, M. Optimal time-jerk trajectory planning for industrial robots. *Mech. Mach. Theory* **2018**, *121*, 530–544. [[CrossRef](#)]
15. Fang, Y.; Hu, J.; Liu, W.; Shao, Q.; Qi, J.; Peng, Y. Smooth and time-optimal S-curve trajectory planning for automated robots and machines. *Mech. Mach. Theory* **2019**, *137*, 127–153. [[CrossRef](#)]
16. Kim, J.; Croft, E.A. Online near time-optimal trajectory planning for industrial robots. *Robot. Comput. Integr. Manuf.* **2019**, *58*, 158–171. [[CrossRef](#)]
17. Zhang, L.; Wang, Y.; Zhao, X.; Zhao, P.; He, L. Time-optimal trajectory planning of serial manipulator based on adaptive cuckoo search algorithm. *J. Mech. Sci. Technol.* **2021**, *35*, 3171–3181. [[CrossRef](#)]
18. Luo, L.P.; Yuan, C.; Yan, R.J.; Yuan, Q.; Wu, J.; Shin, K.S.; Han, C.S. Trajectory planning for energy minimization of industry robotic manipulators using the Lagrange interpolation method. *Int. J. Precis. Eng. Manuf.* **2015**, *16*, 911–917. [[CrossRef](#)]
19. Liu, Z.; Xu, J.; Cheng, Q.; Zhao, Y.; Pei, Y.; Yang, C. Trajectory Planning with Minimum Synthesis Error for Industrial Robots Using Screw Theory. *Int. J. Precis. Eng. Manuf.* **2018**, *19*, 183–193. [[CrossRef](#)]
20. Bakshi, S.; Feng, T.; Yan, Z.; Ma, Z.; Chen, D. Energy-Conscientious Trajectory Planning for an Autonomous Mobile Robot in an Asymmetric Task Space. *J. Intell. Robot. Syst. Theory Appl.* **2021**, *101*, 1–14. [[CrossRef](#)]
21. Lin, H.I. A fast and unified method to find a minimum-jerk robot joint trajectory using particle swarm optimization. *J. Intell. Robot. Syst. Theory Appl.* **2014**, *75*, 379–392. [[CrossRef](#)]
22. Ma, J.; Gao, S.; Yan, H.; Lv, Q.; Hu, G. A new approach to time-optimal trajectory planning with torque and jerk limits for robot. *Rob. Auton. Syst.* **2021**, *140*, 1–10. [[CrossRef](#)]
23. Dai, C.; Lefebvre, S.; Yu, K.M.; Geraedts, J.M.P.; Wang, C.C.L. Planning Jerk-Optimized Trajectory with Discrete Time Constraints for Redundant Robots. *IEEE Trans. Autom. Sci. Eng.* **2020**, *17*, 1711–1724. [[CrossRef](#)]
24. Duan, H.; Zhang, R.; Yu, F.; Gao, J.; Chen, Y. Optimal Trajectory Planning for Glass-Handling Robot Based on Execution Time Acceleration and Jerk. *J. Robot.* **2016**, *2016*, 1–10. [[CrossRef](#)]

25. Chen, D.; Li, S.; Wang, J.F.; Feng, Y.; Liu, Y. A multi-objective trajectory planning method based on the improved immune clonal selection algorithm. *Robot. Comput. Integr. Manuf.* **2019**, *59*, 431–442. [[CrossRef](#)]
26. Yin, S.; Ji, W.; Wang, L. A machine learning based energy efficient trajectory planning approach for industrial robots. *Procedia CIRP* **2019**, *81*, 429–434. [[CrossRef](#)]
27. Zhang, X.; Huang, Y.; Rong, Y.; Li, G.; Wang, H.; Liu, C. Optimal trajectory planning for wheeled mobile robots under localization uncertainty and energy efficiency constraints. *Sensors* **2021**, *21*, 335. [[CrossRef](#)]
28. Chang, Z.; Hao, L.; Yan, Q.; Ye, T. Research on Manipulator Tracking Control Algorithm Based on RBF Neural Network. *IOP Conf. Ser. Earth Environ. Sci.* **2021**, *1802*, 1–8. [[CrossRef](#)]
29. Liu, Q.; Li, D.; Ge, S.S.; Ji, R.; Ouyang, Z.; Tee, K.P. Adaptive bias RBF neural network control for a robotic manipulator. *Neurocomputing* **2021**, *447*, 213–223. [[CrossRef](#)]
30. Lin, C.J.; Sie, T.Y.; Chu, W.L.; Yau, H.T.; Ding, C.H. Tracking control of pneumatic artificial muscle-activated robot arm based on sliding-mode control. *Actuators* **2021**, *10*, 66. [[CrossRef](#)]
31. Yeh, Y.-L. A Robust Noise-Free Linear Control Design for Robot Manipulator with Uncertain System Parameters. *Actuators* **2021**, *10*, 121. [[CrossRef](#)]
32. Ayeb, A.; Chatti, A. Sliding Mode Control of Nonholonomic Uncertain Perturbed Wheeled Mobile Robot. *Int. J. Robot. Autom.* **2021**, *36*. [[CrossRef](#)]
33. Al-Darraj, I.; Piromalis, D.; Kakei, A.A.; Khan, F.Q.; Stojmenovic, M.; Tsaramiris, G.; Papageorgas, P.G. Adaptive robust controller design-based rbf neural network for aerial robot arm model. *Electronics* **2021**, *10*, 831. [[CrossRef](#)]
34. Xu, N.; Zhao, X.; Zong, G.; Wang, Y. Adaptive control design for uncertain switched nonstrict-feedback nonlinear systems to achieve asymptotic tracking performance. *Appl. Math. Comput.* **2021**, *408*, 126344. [[CrossRef](#)]
35. Atique, M.M.U.; Sarker, M.R.I.; Ahad, M.A.R. Development of an 8DOF quadruped robot and implementation of Inverse Kinematics using Denavit-Hartenberg convention. *Heliyon* **2018**, *4*, 1–19. [[CrossRef](#)]
36. Wang, F.; Chao, Z.Q.; Huang, L.B.; Li, H.Y.; Zhang, C.Q. Trajectory tracking control of robot manipulator based on RBF neural network and fuzzy sliding mode. *Clust. Comput.* **2019**, *22*, 5799–5809. [[CrossRef](#)]
37. Messaoudi, A.; Sadaka, R.; Sadok, H. Matrix recursive polynomial interpolation algorithm: An algorithm for computing the interpolation polynomials. *J. Comput. Appl. Math.* **2020**, *373*, 112471. [[CrossRef](#)]
38. Sheng, G.; Gao, G.; Zhang, B. Application of improved wavelet thresholding method and an RBF network in the error compensating of an MEMS gyroscope. *Micromachines* **2019**, *10*, 608. [[CrossRef](#)] [[PubMed](#)]
39. Wang, L.; Zhou, X.; Hu, T. A new computed torque control system with an uncertain rbf neural network controller for a 7-dof robot. *Teh. Vjesn.* **2020**, *27*, 1492–1500. [[CrossRef](#)]
40. Gao, L.; Xiong, L.; Lin, X.; Xia, X.; Liu, W.; Lu, Y.; Yu, Z. Multi-sensor fusion road friction coefficient estimation during steering with lyapunov method. *Sensors* **2019**, *19*, 3816. [[CrossRef](#)] [[PubMed](#)]
41. Wang, H.; Fang, L.; Song, T.; Xu, J.; Shen, H. Model-free adaptive sliding mode control with adjustable funnel boundary for robot manipulators with uncertainties. *Rev. Sci. Instrum.* **2021**, *92*, 1–10. [[CrossRef](#)] [[PubMed](#)]