robotics

MDPI

Article

Improving the Accuracy of a Robot by Using Neural Networks (Neural Compensators and Nonlinear Dynamics)

Zhengjie Yan [1], Yury Klochkov [1] and Lin Xi [2,*]

[1] Institute of Computer Science and Technology, Peter the Great St. Petersburg Polytechnic University, Polytechnicheskaya 29, St. Petersburg 195251, Russia
[2] Information Construction Management Office, MinZu University of China, No. 27 South Street, Zhongguancun, Haidian District, Beijing 100081, China
* Correspondence: xilin@muc.edu.cn

Abstract: The subject of this paper is a programmable con trol system for a robotic manipulator. Considering the complex nonlinear dynamics involved in practical applications of systems and robotic arms, the traditional control method is here replaced by the designed Elma and adaptive radial basis function neural network—thereby improving the system stability and response rate. Related controllers and compensators were developed and trained using MATLAB-related software. The training results of the two neural network controllers for the robot programming trajectories are presented and the dynamic errors of the different types of neural network controllers and two control methods are analyzed.

Keywords: robot manipulator; programmable control system; neural network; nonlinear multivariate compensators; modeling; dynamic analysis; dynamic errors

check for updates

Citation: Yan, Z.; Klochkov, Y.; Xi, L. Improving the Accuracy of a Robot by Using Neural Networks (Neural Compensators and Nonlinear Dynamics). *Robotics* **2022**, *11*, 83. https://doi.org/10.3390/robotics11040083

Academic Editor: Antonio Paulo Moreira

Received: 16 July 2022
Accepted: 17 August 2022
Published: 19 August 2022

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

Copyright: © 2022 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

In the early control design period of the manipulator, the dynamic model of the system and related system parameters need to be accurately described when designing the controller [1]. In traditional control design methods, such as computational torque control and inverse dynamics control, which works fine [2], by calculating the torque of the robot arm and the construction of your dynamic equation, you can get a good control effect [3]. However, this is under the premise of being able to get an accurate data model. However, it is difficult to obtain an accurate mathematical model of a robot during its actual production and use [4]. Furthermore, due to the effects of different payloads, it may be difficult to obtain appropriate model-based methods. Recently, neural network calculators have been used to improve the characteristics of robotic manipulator control systems in the development of robotic manipulator control systems. In numerical control (CNC) systems, a neural network interpolator of robot link trajectories can be used to replace the traditional spline interpolator [5].

This research is used to train compensators using neural networks in numerical control systems of robotic manipulators and in the absence of precise initial data [6]. The adaptive neural network compensator is used to replace the traditional PID controller and other methods for compensating the dynamic error caused by the torsional load in the robot link drive [7]. The nonlinear dynamic coupling of the drive selects the two-strait robot manipulator in the angular coordinate system as the simulated control object [8].

The purpose of this work is to synthesize and train a multi-dimensional neural network controller to compensate and correct the dynamic error of the robot trajectory. Both the neural network controller and simulation of the project were performed in MATLAB(R2018b, The MathWorks, Inc. Protected by U.S and international patents) [9].

There are mainly two types of neural networks trained in this paper: an Elman neural network and an RBF adaptive neural network.

An Elman neural network is a typical local regression network (global feed forward local recurrent). An Elman network can be viewed as a recurrent neural network with local memory units and local feedback connections [10]. Its main structure is a feedforward connection, including an input layer, hidden layer, and output layer, and its connection weight can be modified by learning; the feedback connection is composed of a group of "structural" units that are used to memorize the output value of the previous moment. The connection weights are fixed. In this kind of network, in addition to the ordinary hidden layer, there is a special hidden layer called the association layer (or connection unit layer) [11]; this layer receives feedback signals from the hidden layer and each hidden layer node—each has a corresponding association layer node connection. The role of the association layer is to use the state of the hidden layer at the previous moment together with the network input at the current moment as the input of the hidden layer through the connection memory, which is equivalent to state feedback [12]. It is a dynamic feedback network, which can internally feedback, store, and utilize the output information of the previous moment. It can not only realize the modeling of the static system, but also realize the mapping of the dynamic system and directly reflect the dynamic characteristics of the system. It is better than a BP neural network in terms of its performance and stability [13].

The structure of the RBF network is similar to the multi-layer forward network, which is a three-layer forward network. The input layer is composed of signal source nodes; the second layer is the hidden layer. The number of hidden units depends on the needs of the described problem. The transformation function of the hidden unit is the RBF radial basis function, which is radially symmetric to the center point. It has a decaying non-negative nonlinear function; the third layer is the output layer, which responds to the effects of the input mode [14]. The transformation from the input space to the hidden layer space is nonlinear, while the transformation from the hidden layer space to the output layer space is linear. This also makes the RBF neural network have a very fast system response speed, which is also its biggest feature [15].

In this paper, by constructing the dynamic equation of the manipulator robotic, we tested the two constructed neural networks in the same state and obtained the following results: 1. The two constructed neural networks reached the theoretical position in the case of unknown perturbations and incomplete dynamic model data and demonstrated the high efficiency of the control. 2. Through the results of the external disturbance compensation, it was also found that although the response speed of the RBF adaptive neural network was faster, its accuracy was lower than that of the Elman neural network. However, the RBF adaptive neural network performed better in local approximations. 3. For the designed RBF adaptive neural network, the control algorithm was directly designed in the workspace, which improved the system response time and was able to ensure that the adaptive control was always stable.

2. Materials and Methods

In this paper, the controlled object was selected as the n-joint manipulator, and its dynamic equation was:

$$M(q)\ddot{q} + C\left(q,\dot{q}\right)\dot{q} + G(q) = \tau - \tau_{\mathrm{d}} \tag{1}$$

where $M(q)$ is the $n \times n$ positive definite inertia matrix of the order; $C\left(q,\dot{q}\right)$ is for the $n \times n$ order centrifugation and Gohrenheit force terms; $G(q)$ is an $n \times 1$ order gravity term; the q vectors are the joint variables; τ is the moment acting on the joint; and τ_{d} is the disturbance for the outside world. In practical engineering, $M(q)$, $C\left(q,\dot{q}\right)$, and $G(q)$ are often unknown factors and can be expressed as follows:

$$\begin{aligned} M(q) &= M_0(q) + E_{\mathrm{M}} \\ C\left(q,\dot{q}\right) &= C_0\left(q,\dot{q}\right) + E_{\mathrm{C}} \\ G(q) &= G_0(q) + E_{\mathrm{G}} \end{aligned} \tag{2}$$

where E_M, E_C, and E_G are modeled errors for $M(q)$, $C(q,\dot{q})$, and $G(q)$, respectively. The controller design defines the tracking error as:

$$e(t) = q_d(t) - q(t) \tag{3}$$

where $q_d(t)$ is the ideal tracking instructions and $q(t)$ is the actual location. The sliding mode function is defined as:

$$r = \dot{e} + \Lambda e \tag{4}$$

where $\Lambda > 0$. $\dot{q}_r = r(t) + \dot{q}(t)$ is defined, followed by $\ddot{q}_r = \dot{r}(t) + \ddot{q}(t)$, $\dot{q}_r = \dot{q}_d + \Lambda e$ and $\ddot{q}_r = \ddot{q}_d + \Lambda \dot{e}$ by Equation (1), as follows:

$$\begin{aligned} \tau &= M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + \tau_d \\ &= M(q)(\ddot{q}_r - \dot{r}) + C(q,\dot{q})(\dot{q}_r - r) + G(q) + \tau_d \\ &= M(q)\ddot{q}_r + C(q,\dot{q})\dot{q}_r + G(q) - M(q)\dot{r} - C(q,\dot{q})r + \tau_d \\ &= M_0(q)\ddot{q}_r + C_0(q,\dot{q})\dot{q}_r + G_0(q) + E' - M(q)\dot{r} - C(q,\dot{q})r + \tau_d \end{aligned} \tag{5}$$

Among these $E' = E_M\ddot{q}_r + E_C\dot{q}_r + E_G$ for the above system and the design controller is as follows:

$$\tau = \tau_m + K_p r + K_i \int r dt + \tau_r \tag{6}$$

where $K_p > 0; K_i > 0; \tau_m$ for the control term based on the nominal model; τ_r is for robust terms and

$$\begin{aligned} \tau_m &= M_0(q)\ddot{q}_r + C_0(q,\dot{q})\dot{q}_r + G_0(q) \\ \tau_r &= K_r \mathrm{sgn}(r) \end{aligned} \tag{7}$$

where $K_r = \mathrm{diag}[k_{rii}]; k_{rii} \geqslant |E_i|, i = 1, \cdots, n; E = E' + \tau_d$ is the coefficient matrix of the linear compensator. The purpose of this project was to design a controller through the Elman and RBF adaptive neural networks. Through simulation and training, the unknown external disturbance can be compensated more quickly, so as to obtain higher stability and shorten the system response time.

By synthesizing Equations (5)–(7), we can obtain:

$$\begin{aligned} &M_0(q)\ddot{q}_r + C_0(q,\dot{q})\dot{q}_r + G_0(q) - M(q)\dot{r} - C(q,\dot{q})r + E\prime + \tau_r \\ &= M_0(q)\ddot{q}_r + C_0(q,\dot{q})\dot{q}_r + G_0(q) + K_p r + K_i \textstyle\int_0^t r dt + K_r \mathrm{sgn}(r) \end{aligned} \tag{8}$$

where

$$M(q)\dot{r} + C(q,\dot{q})r + K_i \int_0^t r dt = -K_p r - K_r \mathrm{sgn}(r) + E \tag{9}$$

Under this model, the traditional controller can only work under the unknown model and uncertainty, which causes the control input to chatter. To solve this problem, two kinds of neural networks were used to approximate the model and compensate for external disturbances.

3. Training of Nonlinear Neural Network Compensators

3.1. Designing Compensators with Elman Neural Networks

This part of the article is for the development of a model of an Elman neural control scheme for nonlinear dynamic systems in a manipulative robot.

As an example, consider an industrial n-link robot manipulator, the links of which are interconnected by rotational motion drives. The position of the links is determined by the angles φ_1, φ_2, φ_2 ... In addition, weight forces act on the robot links, which are directed at a certain angle α to the selected coordinate system, which demonstrates the ability of the device to work at any angle to the horizon [16].

The object of the control is the n joint robotic arm, whose kinetic equation is:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + F(\dot{q}) + \tau_d = \tau \tag{10}$$

Through the dynamic analysis of the manipulator in the second part and Equations (1)–(4), the following equations can be obtained:

$$\begin{aligned} M\dot{r} &= M\left(\ddot{q}_{\mathrm{d}} - \ddot{q} + \Lambda\dot{e}\right) = M\left(\ddot{q}_{\mathrm{d}} + \dot{e}\right) - M\ddot{q} \\ &= M\left(\ddot{q}_{\mathrm{d}} + \Lambda\dot{e}\right) + C\dot{q} + G + F + \tau_{\mathrm{d}} - \tau \\ &= M\left(\ddot{q}_{\mathrm{d}} + \Lambda\dot{e}\right) - Cr + C\left(\dot{q}_{\mathrm{d}} + \Lambda e\right) + G + F + \tau_{\mathrm{d}} - \tau \\ &= -Cr - \tau + f + \tau_{\mathrm{d}} \end{aligned} \tag{11}$$

where $f(x) = M\ddot{q}_{\mathrm{r}} + C_{\mathrm{r}} + G + F; \dot{q}_{\mathrm{r}} = \dot{q}_{\mathrm{d}} + \Lambda e$. As $f(x)$, you can see from the expression $f(x)$ that it contains all the model information—that is, in Equation (7), all the model information can be represented by $f(x)$. The control goal was to use the neural network approximation $f(x)$ to design a robust controller that does not require model information. The approximation used was Elman network approximation $f(x)$ and the network algorithm was:

$$\begin{aligned} h_j &= x_j(k-1),\ j = 1, 2, \cdots, m \\ f(x) &= W1^{\top} h(k) + \mathrm{W2}(\mathrm{u}(k-1)) + \varepsilon \end{aligned} \tag{12}$$

where x is the input of the Elman neural network; W is the ideal weights for the network;

W1 connect the weights from the input layer to the middle layer; W2 connects the weights from the output layer to the middle layer $h = \begin{bmatrix} h_1 & h_2 & \cdots & h_m \end{bmatrix}^{\mathrm{T}}$; ε is a very small, positive real number. Approximation was carried out using Elman neural networks $f(x)$; j is the number of neurons in the hidden layer of the neural network.

The advantage of the Elman neural network is increased stability, since it has feedbacks from the outputs of internal neurons to the intermediate layer, which makes it more stable compared to a recurrent network of a similar type (for example, the Hopfield neural network, in which internal feedbacks are brought to the primary inputs, where signals are mixed). In addition, the Elman neural network allows you to take into account the background of the observed processes and accumulate information to choose the right robot control strategy [11].

$$\widehat{f}(x) = \widehat{W}1^{\top} h(k) + \widehat{W}2(\mathrm{u}(k-1))$$

where $\widetilde{W} = W - \widehat{W}; \parallel W \parallel_F \leqslant W_{\max}$. Combining the above equation and Formula (12), we can get:

$$\begin{aligned} f - \widehat{f} &= W1^{\top} h(k-1) + \mathrm{W2}(k) + \varepsilon - \widehat{W}1^{\top} h(k) + \widehat{W}2(\mathrm{u}(k-1)) \\ &= \widetilde{W}1^{\top} h(k-1) + \widetilde{W}2(k) + \varepsilon \end{aligned} \tag{13}$$

By Equation (13), referencing the [11] design methods, the design control laws are:

$$\tau = \widehat{f}(x) + K_{\mathrm{v}} r - v \tag{14}$$

where $v = -(\varepsilon_{\mathrm{N}} + b_{\mathrm{d}})\mathrm{sgn}(r)$ for the robust term; $\widehat{f}(x)$ is an approximation of $f(x)$. ELMAN The network weight adaptive law is:

$$\dot{\widehat{W}} = \Gamma h r^{\mathrm{T}} \tag{15}$$

where $\Gamma = \Gamma^{\mathrm{T}} > 0$. The general Equation (13) can be substituted into (10), a collation of the Elman neural network adaptive control MATLAB simulation:

$$\begin{aligned} M\dot{r} &= -Cr - \left(\widehat{f}(x) + K_{\mathrm{v}} r - v\right) + f + \tau_{\mathrm{d}} \\ &= -(K_{\mathrm{v}} + C)r + \widetilde{W}1^{\top} h(k) + \widetilde{W}2(\mathrm{u}(k-1)) + \varepsilon + (\varepsilon + \tau_{\mathrm{d}}) + v \\ &= -(K_{\mathrm{v}} + C)r + \zeta_1 \end{aligned} \tag{16}$$

where $\zeta_1 = \widetilde{W}1^{\mathrm{T}}(k)\varphi + W2\left(\mathrm{u}\left(\widetilde{k}-1\right)\right) + (\varepsilon + \tau_{\mathrm{d}}) + v$; with reference by [7], the analysis of the closed-loop system is as follows: the Lyapunov function is defined as:

$$L = \frac{1}{2}r^{\mathrm{T}}Mr + \frac{1}{2}\mathrm{tr}\left(\widetilde{W}1(k-1)^{\mathrm{T}}\Gamma^{-1}\widetilde{W1(k)}\right) + \frac{1}{2}\mathrm{tr}\ \mathrm{u}\left(W2\left(\widetilde{k}-1\right)\right) \tag{17}$$

Derivating Equation (17) to get:

$$\dot{L} = r^{\mathrm{T}}M\dot{r} + \frac{1}{2}r^{\mathrm{T}}\dot{M}r + \mathrm{tr}\left(\widetilde{\dot{\widehat{W}}1(k)}^{\mathrm{T}}\Gamma^{-1}\dot{\overline{W1}}\right) + \mathrm{tru}\left(\widetilde{\dot{\widehat{W}}2(k-1)}^{\mathrm{T}}\Gamma^{-1}\dot{\overline{W2}}\right)$$

By substituting Equation (15) into the above equation, the following Equation can be defined:

$$\begin{aligned}\dot{L} = \quad & -r^{\mathrm{T}}K_{\mathrm{v}}r + \tfrac{1}{2}r^{\mathrm{T}}\left(\dot{M}-2C\right)r \\ & +\mathrm{tr}\widetilde{W1(k)}^{\mathrm{T}}\left(\Gamma^{-1}\dot{W}1 + hr^{\mathrm{T}}\right) + \mathrm{tr}W2(k-1)\left(\Gamma^{-1}\dot{W}2\right) + r^{\mathrm{T}}(\varepsilon + \tau_{\mathrm{d}} + v)\end{aligned} \tag{18}$$

According to the following conditions:

(1) oblique symmetry characteristics of the manipulator $r^{\mathrm{T}}\left(\dot{M}-2C\right)r = 0$;

(2) $r^{\mathrm{T}}\widetilde{W}^{\mathrm{T}}h = \mathrm{tr}\left(W^{\mathrm{T}}hr^{\mathrm{T}}\right)$;

(3) $\dot{\overrightarrow{W}} = -\dot{\widehat{W}} = -\Gamma hr^{\mathrm{T}}$.

where

$$\dot{L} = -r^{\mathrm{T}}K_{\mathrm{v}}r + r^{\mathrm{T}}(\varepsilon + \tau_{\mathrm{d}} + v) \tag{19}$$

consider

$$\begin{aligned} r^{\mathrm{T}}(\varepsilon + \tau_{\mathrm{d}} + v) \quad & = r^{\mathrm{T}}(\varepsilon + \tau_{\mathrm{d}}) + r^{\mathrm{T}}(-(\varepsilon_{\mathrm{N}} + b_{\mathrm{d}})\mathrm{sgn}(r)) \\ & = r^{\mathrm{T}}(\varepsilon + \tau_{\mathrm{d}}) - \parallel r \parallel (\varepsilon_{\mathrm{N}} + b_{\mathrm{d}}) \leqslant 0 \end{aligned} \tag{20}$$

then the following definition can be obtained

$$\dot{L} \leqslant -r^{\mathrm{T}}K_{\mathrm{v}}r \leqslant 0 \tag{21}$$

The neural network learning algorithm consists of the following steps:

1. At the initial moment of time t = 0, all neurons of the hidden layer are set to the zero position—the initial value is zero.
2. The input value is fed to the network, where it is directly distributed.
3. Set t = t + 1 and make the transition to step 2; neural network training is performed until the total root mean-square error of the network takes the smallest value.

3.2. Designing a Compensator with an Adaptive Radial Basis Function Neural Network for Local Model Approximation

In this section, a nonlinear dynamic model compensator is designed based on an Adaptive Radial basis function (RBF) neural network, based on the literature [8,11], which is then compared with the neural network compensator designed in the previous section.

The dynamic equation of the manipulator has the following properties:

Property 1: The inertia matrix $M_x(q)$ is symmetric positive definite;

Property 2: If $C_x(q,\dot{q})$ is defined by Christoffel's notation rule, the matrix $\dot{M}_x(q) - 2C_x(q,\dot{q})$ is skew symmetric [17].

Since $M_x(q)$ and $G_x(q)$ are just functions of q , they can be modeled using static neural networks [16,18–20].

$$\begin{aligned} m_{xkj}(q) &= \sum_l \theta_{kij}\xi_{kij}(q) + \varepsilon_{mkj}(q) = \theta_{kj}^{\mathrm{T}}\xi_{kj}(q) + \varepsilon_{mkj}(q) \\ g_{xk}(q) &= \sum_l \beta_{kl}\eta_{kl}(q) + \varepsilon_{kk}(q) = \beta_k^{\mathrm{T}}\eta_k(q) + \varepsilon_{kk}(q) \end{aligned} \tag{22}$$

Among these, $\theta_{kjl}, \beta_{kl} \in R$ are the weights of the neural network; $\xi_{kjl}(q), \eta_{kl}(q) \in R$ is the radial basis function whose input is a vector; q, $\varepsilon_{mkj}(q), \varepsilon_{kk}(q) \in R$ are the modeling errors of $m_{xkj}(q)$ and $g_{xk}(q)$, respectively, and are assumed to be bounded.

For $C(q, \dot{q})$, modeling with a dynamic neural network with inputs q and q, the neural network model of $C_{xkj}(q, \dot{q})$ is:

$$C_{xkj}(q, \dot{q}) = \sum_l \alpha_{kil}\xi_{kil}(z) + \varepsilon_{ikj}(z) = \alpha_{kj}^{\mathrm{T}}\xi_{kj}(z) + \varepsilon_{ikj}(z) \tag{23}$$

Among these, $z = \begin{bmatrix} q^{\mathrm{T}} & \dot{q}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \in R^{2n}$; $\alpha_{kjl} \in R$ is the weight; $\xi_{kjl}(z) \in R$ is the radial basis function of the input vector z; and $\varepsilon_{ckj}(z)$ is the modeling error of the element $c_{xkj}(q, \dot{q})$, assuming it is also bounded [20,21].

Using neural network modeling, the dynamic equation of the manipulator in space can be written as:

$$M_x(q)\ddot{x} + C_x(q, \dot{q})\dot{x} + G_x(q) = F_x \tag{24}$$

Using GL (general linear) matrices and their multiplication operations, $M_x(q)$ can be written as:

$$M_x(q) = \left[\{\theta\}^{\top} \cdot \{\Xi(q)\}\right] + E_{\mathrm{M}}(q) \tag{25}$$

where {θ} and {Ξ(q)} are GL (general linear) matrices whose elements are θ_{kj}^{T} and $\xi_{kj}(q)$; E_M (q)∈Rˆ(n × n) is a matrix whose elements are modeling errors $\varepsilon_{mkj}(q)$.

Similarly, for $C(q, \dot{q})$ and $G_x(q)$:

$$C_x(q, q\,.) = \left[\{A\}^{T} \cdot \{Z(z)\}\right] + E_C(z) \quad G\_x\,(q) = [\{B\}T \cdot \{H(q)\}] + E\_G\,(q)) \tag{26}$$

where $\{A\}, \{Z(z)\}$, $\{B\}$, and $\{H(q)\}$ are GL matrices and GL vectors whose elements are α_{kj}, $\xi_{kj}(z)$, β_k and $\eta_k(q)$; $E_{\mathrm{C}}(z) \in R^{n \times n}$ and $E_{\mathrm{G}}(q) \in R^n$ are the elements and matrices of the modeling errors $\varepsilon_{ckj}(z)$ and $\varepsilon_{kk}(q)$, respectively.

Assuming $x_d(t)$ is the ideal trajectory of the workspace, then $\dot{x}_d(t)$ and $\ddot{x}_d(t)$ are the ideal velocity and ideal acceleration:

$$\begin{aligned} \dot{x}_{\mathrm{r}}(t) &= \dot{x}_{\mathrm{d}}(t) + \Lambda e(t) \\ r(t) &= \dot{x}_{\mathrm{r}}(t) - \dot{x}(t) = \dot{e}(t) + \Lambda e(t) \end{aligned} \tag{27}$$

where Λ is a positive definite matrix.

Lemma 1 (Barbalat's Lemma). *If the function $h : R \to R$ is a uniform continuous function defined by $[0, +\infty)$, and $lim_{t\to\infty} \int_0^t h(\delta)d$ exists and is finite, then $lim_{t\to\infty} h(t) = 0$ is obtained* [22].

Lemma 2. *Let $e(t) = h(t) * r(t)$, where * represents convolution, $h(t) = L^{-1}H(s)$ and H(s) is a strictly exponentially stable transfer function of order n * n. If $r \in L_2^n$, then $e \in L_2^n \cap L_\infty^n, \dot{e} \in L_2^n, e$ is continuous, $e \to 0, r \to 0, \dot{e} \to 0$ when $t \to \infty$.*

Consider a second-order SISO system and let $r(t) = ce(t) + \dot{e}(t)$, then we can obtain $r(s) = e(s)(c + s)$, $e(s) = \frac{1}{s+c}r(s)$, and $H(s) = \frac{1}{s+c}$. To ensure that $H(s)$ is a strictly exponentially stable transfer function, it can be defined that c > 0. If the above conditions are satisfied, then $r(t) = 0$, ce(t) + e(t) = 0 can be obtained, thus ensuring that the system is exponentially closed [17].

Using $(\widehat{\sigma})$ to represent the estimated value of $(\cdot)$, define $d(\cdot) = (\cdot) - (\widehat{\sigma})$, then $\left\{\widehat{\theta}\right\}$, $\left\{\widehat{A}\right\}$, and $\left\{\widehat{B}\right\}$ represents the $\{\theta\}$, $\{A\}$, and $\{B\}$ estimates.

Designing the controller this can be defined as:

$$F_x = \left[\left\{\widehat{\theta}\right\}^{\mathrm{T}}\cdot\{\theta(q)\}\right]\ddot{x}_{\mathrm{r}} + \left[\left\{\widehat{A}\right\}^{\mathrm{T}}\cdot\{Z(z)\}\right]\dot{x}_{\mathrm{r}} + \left[\left\{\widehat{B}\right\}^{\mathrm{T}}\cdot\{H(q)\}\right] + Kr + k_{\mathrm{s}}\mathrm{sgn}(r) \quad (28)$$

where $K \in R^{n\times n} > 0; k_{\mathrm{s}} > \| E \|; E = E_{\mathrm{M}}(q)\ddot{x}_{\mathrm{r}} + E_{\mathrm{C}}(z)\dot{x}_{\mathrm{r}} + E_{\mathrm{G}}(q)$. The first three terms of the controller are model-based controls, the K_r term is equivalent to the proportional derivative (PD) control, and the last term of the control law is a robust term that suppresses the modeling error of the neural network.

From the expression of the controller, it is obvious that the controller does not need to solve the Jacobian inverse matrix. In actual control, the loser can be obtained by $\tau = J^{\mathrm{T}}(q)F_x$ [19].

Substituting Equations (24) and (25) into Equation (23), we can obtain:

$$\begin{array}{l}\left\{\left[\{\theta\}^{\mathrm{T}}\cdot\{\Xi(q)\}\right] + E_{\mathrm{M}}(q)\right\}\ddot{x} + \left\{\left[\{A\}^{\mathrm{T}}\cdot\{Z(z)\}\right] + E_{\mathrm{C}}(z)\right\}\dot{x} \\ +\left[\{B\}^{\mathrm{T}}\cdot\{H(q)\}\right] + E_{\mathrm{G}}(q) = F_x\end{array} \quad (29)$$

Substituting the control law (27) into the above formula, we can obtain:

$$\begin{array}{rl} & \left\{\left[\{\theta\}^{\mathrm{T}}\cdot\{E(q)\}\right] + E_{\mathrm{M}}(q)\right\}\ddot{x} + \left\{\left[\{A\}^{\mathrm{T}}\cdot\{Z(z)\}\right]\right. \\ & \left.+E_{\mathrm{C}}(z)\right\}\dot{x} + \left[\{B\}^{\mathrm{T}}\cdot\{H(q)\}\right] + E_{\mathrm{G}}(q) \\ = & \left[\left\{\widehat{\theta}\right\}^{\mathrm{T}}\cdot\{\Xi(q)\}\right]\ddot{x}_{\mathrm{r}} + \left[\left\{\widehat{A}\right\}^{\mathrm{T}}\cdot\{Z(z)\}\right]\dot{x}_{\mathrm{r}} \\ & +\left[\left\{\widehat{B}\right\}^{\mathrm{T}}\cdot\{H(q)\}\right] + Kr + k_{\mathrm{s}}\mathrm{sgn}(r)\end{array} \quad (30)$$

Substituting X = $\dot{x}_{\mathrm{r}} - r$ and $\ddot{x}_{\mathrm{r}} = \ddot{x}_{\mathrm{r}} - \dot{r}$ into the above equation, we can obtain:

$$\begin{array}{rl} & \left\{\left[\{\theta\}^{\mathrm{T}}\cdot\{\theta(q)\}\right] + E_{\mathrm{M}}(q)\right\}\left(\ddot{x}_{\mathrm{r}} - \dot{r}\right) + \left\{\left[\{A\}^{\mathrm{T}}\cdot\{Z(z)\}\right]\right. \\ & \left.+E_{\mathrm{C}}(z)\right\}\left(\dot{x}_{\mathrm{r}} - r\right) + \left[\{B\}^{\mathrm{T}}\cdot\{H(q)\}\right] + E_{\mathrm{G}}(q) \\ = & \left[\left\{\widehat{\theta}\right\}^{\mathrm{T}}\cdot\{\theta(q)\}\right]\ddot{x}_{\mathrm{r}} + \left[\left\{\widehat{A}\right\}^{\mathrm{T}}\cdot\{Z(z)\}\right]\dot{x}_{\mathrm{r}} + \left[\left\langle\widehat{B}\right\rangle^{\top}\cdot\{H(q)\right\rangle] + Kr + k_s\mathrm{sgn}(r)\end{array} \quad (31)$$

Substituting Equations (24) and (25) into the above equations, we can obtain:

$$\begin{array}{rl} & M_f(q)\dot{r} + C_x\left(q,\dot{q}\right)r + Kr + k_s\mathrm{sgn}(r) \\ = & \left[\{\theta\}^{\top}\cdot\langle B(q)\}\right]\ddot{x}_{\mathrm{r}} + \left[\{\widehat{A}\rangle^{\top}\cdot\{Z(z)\}\right]\dot{x}_{\mathrm{r}} + \left[\{\widetilde{B}\}^{\top}\cdot\{H(q)\}\right] + E\end{array} \quad (32)$$

For the closed-loop system, if $K > 0, k, > \| E \|$ parallel, and the adaptive law is designed as:

$$\begin{array}{ll}\dot{\widehat{\theta}}_k & = \Gamma_k\cdot\{\xi_k(q)\}\ddot{x}_t r_k \\ \dot{\widehat{\alpha}}_k & = Q_k\cdot\{\xi_k(z)\}\dot{x}_r r_k \\ \dot{\widehat{\beta}}_k & = N_{\downarrow}\eta_k(q)r_k\end{array} \quad (33)$$

Among these, $\Gamma_k = \Gamma_k^T > 0; Q_k = Q_k^T > 0; N_k = N_k^T > 0$ and $\widehat{\theta}_k$ and $\widehat{\alpha}_k$ are the vectors of $\widehat{\theta}_k$ and $\widehat{a}_k$, respectively; then, $\widehat{\theta}_k, \widehat{\alpha}_t\cdot\widehat{\beta}_k \in L\ldots, e \in L_2^* \cap L^*, e$ is continuous. And $e \to 0$ and $\dot{e} \to 0$ when $t \to \infty$.

According to an integral line Lyapunov function proposed in Reference, the stability can be analyzed as follows:

$$RV = \frac{1}{2}r^{\mathrm{T}}M_k(q)r + \frac{1}{2}\sum_{k=1}^{n}\theta_k^{\mathrm{T}}\Gamma_k^{-1}\theta_k + \frac{1}{2}\sum_{k=1}^{n}\widetilde{\alpha}_k^{\mathrm{T}}Q_k^{-1}\widetilde{\alpha}_k + \frac{1}{2}\sum_{k=1}^{n}\hat{\beta}_k^{\mathrm{T}}N_k^{-1}\hat{\beta}_k \quad (34)$$

Among these, Γ_k, Q_k, and N_k are positive definite pair-second matrices. Derivative with respect to V, we obtain:

$$R\dot{V} = r^{\mathrm{T}} M_k \dot{r} + \frac{1}{2} r^{\mathrm{T}} \dot{M}_k r + \sum_{k=1}^{n} \overline{\theta}_k^{\mathrm{T}} \Gamma_k^{-1} \dot{\theta}_k + \sum_{k=1}^{n} \alpha_k^{\mathrm{T}} Q_k^{-1} \dot{\alpha}_k + \sum_{k=1}^{n} \hat{\beta}_k^{\mathrm{T}} N_k' \dot{\beta}_k \tag{35}$$

Since the matrix $\dot{M}_x(q) - 2C_x(q, \dot{q})$ is obliquely symmetric, then $r^{\mathrm{T}}\left(\dot{M}_x - 2C_x\right) r = 0$, and the below formula can be obtained:

$$\dot{V} = r^{\top}\left(M_s \dot{r} + C_s r\right) - \sum_{k=1}^{n} \overline{\theta}_k^{\top} \Gamma_k^{-1} \dot{\hat{\theta}}_k + \sum_{k=1}^{n} \alpha_l^{\top} Q_k^{-1} \dot{\alpha}_k - \sum_{k=1}^{n} \overline{\beta}_k^{\top} N_k^{-1} \dot{\hat{\beta}}_k \tag{36}$$

Substituting Equation (33) into the above equation, we can obtain:

$$\begin{aligned} \dot{V} = \quad & -r^{\top} K r - k_s r^{\top} \mathrm{sgn}(r) + \sum_{k=1}^{n} \langle \theta_k \rangle^{\top} \cdot \{\xi_k(q)\} \ddot{x}_t r_k + \sum_{k=1}^{n} \tilde{\alpha}_k^{\top} \cdot \langle \xi_k(z) \rangle \dot{x}_* r_k \\ & + \sum_{k=1}^{n} \beta_k^{\top} \eta_k(q) r_k + r^{\top} E - \sum_{k=1}^{n} \theta_k^{\top} \dot{\Gamma}_k' \dot{\theta}_k + \sum_{k=1}^{n} \alpha_k^{\top} Q_k^{-} \dot{\hat{\alpha}}_k - \sum_{k=1}^{n} \hat{\beta}_k^{\top} N_k^{-1} \dot{\hat{\beta}}_k \end{aligned} \tag{37}$$

Substituting the adaptive law (32) into the Equation (37), and combining the inequality $k_n > \| E \|$ parallel, we can obtain:

$$\dot{V} = -r^{\mathrm{T}} K r - k_x r^{\mathrm{T}} \mathrm{sgn}(r) + r^{\mathrm{T}} E \leqslant 0 \tag{38}$$

Convergence Analysis:

(1) From $\dot{V} \leqslant -r^{\top} K r \leqslant 0$ and $K > 0$, from the lemma $e \in L_2^n \cap L^n, \dot{e} \in L_2^n, e$, e∈L2n, e is continuous; then, when $t \to \infty$, $e \to 0, \dot{e} \to 0$

(2) From $\dot{V} \leqslant -r^{\top} K$, we get $0 \leqslant V(t) \leqslant V(0), \forall t \geqslant 0$. Therefore, when $V(t) \in L$, there are $\theta_k, \widehat{\alpha}_k, \beta_k \in L$, and $\widehat{\theta}_k, \widehat{\alpha}_k, \widehat{\beta}_k \in L$.

4. Simulation Study

This section presents the simulation results to evaluate the operation of the proposed adaptive neuro controller. In this section, the proposed control scheme is applied to a two-link project, the mathematical model of which was developed in the Solid works package and imported in the MATLAB package by the second generation of Simulink [22,23].

Initially, the first link moves, but the second one does not. It is shown that after the training of the first link, external disturbances are worked out. Then, the second link starts moving, and the first one stops. After that, both links move, exerting dynamic disturbances on each link. The dynamic equation can be written as:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = \tau - d$$

$$M(q) = \begin{bmatrix} q1 + 2\gamma \cos(q2) & q1 + q2 \cos(q2) \\ q1 + q2 \cos(q2) & q1 \end{bmatrix}$$

$$C(q, \dot{q}) = \begin{bmatrix} -q2\, \dot{q}2 \sin(q2) & -q2(\dot{q}1 + \dot{q}2) \sin(q2) \\ q2\, q1 \sin(q2) & 0 \end{bmatrix}$$

$$G(q) = \begin{bmatrix} g \cos q1 + g \cos(q1 + q2) \\ g \cos(q1 + q2) \end{bmatrix}$$

where the known parameters are j = 12, q1 = 9, q2 = 8, g = 9.8 [23] and the external interferences of the system are $d = d1 + d2e + d\dot{e}$, d1 = 2, d2 = 3, d3 = 6. Assuming that

the expected instructions for tracking the link angle and angular velocity are the following equation:

$$q1d = 1 + 0.2\sin(0.5\pi t)$$

$$q2d = 1 - 0.2\cos(0.5\pi t)$$

then the initial state of the system is [q1 q2]ˆT = [0.6 0.3]ˆT, $\Delta M = 0.6M$, $\Delta C = 0.6C$, $\Delta G = 0.6G$. When modeling, using the formula of the control law and the formula of the adaptive law, $\alpha = 2$, $\gamma = 20, k = 0.001$. In the neural network, the parameters of the Gaussian function were set to c = [−2 −1 0 1 2] and b = 3, and the initial weight was 0.1. The model of the manipulator is shown in Figure 1.

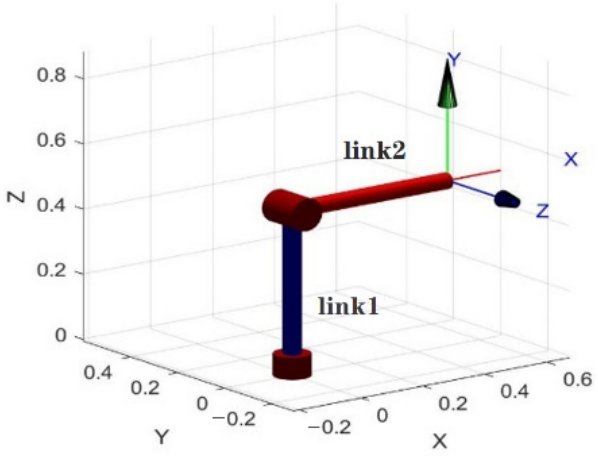

Figure 1. Mechanical model of the robotic manipulator.

The simulation parameters of the robotic arm were as follows:

L(1): 'qlim', [−180 180] * deg; 'm', 0, ... 'Jm', 200×10^{-6}, ... 'G', −62.6111, ... 'B', 1.48×10^{-3}

L(2): 'qlim', [−180 180] * deg; 'm', 17.4, ... 'Jm', 200×10^{-6}, ... 'G', 107.815, ... 'B', 0.817×10^{-3}

(1) Simulation Modeling of Elman Neural Networks

In the Matlab/Simulink system, an artificial neural network model was created to control a manipulative robot, containing an input layer of 15 neurons and a hidden layer varying from 12 to 19 neurons that have local feedback through delay lines.

It can be seen from the Figures 2 and 3 that in the case of adaptive compensation, there was a certain degree of disturbance at the initial stage, and the angular velocity and position tended to converge.

It can be seen from the Figure 4 that the disturbances were compensated and that the robot control process was quite satisfactory.

The experimental results also demonstrate that the proposed method was sufficiently resistant to dynamic perturbations due to unknown situations. The proposed control method is original and successfully exploits the advantages of SMC, neural networks, and adaptive control.

(2) Simulation Modeling of RBF Neural Networks

For the approximation of each element of $M_x(q)$ and $G_x(q)$, it is assumed that the input vector of the RBF neural network is q and the number of hidden layer points is nine. For $C_x(q, \dot{q})$, the input vector $(q, \dot{q})$ of the RBF neural network is assumed and the number of design hidden layer points is seven.

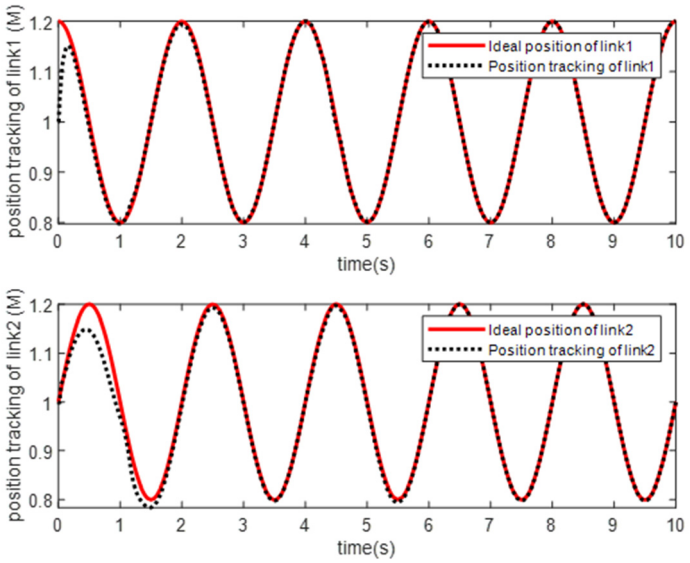

Figure 2. Approximation of the positions of link 1 and link 2 in the case of adaptive Elman compensation.

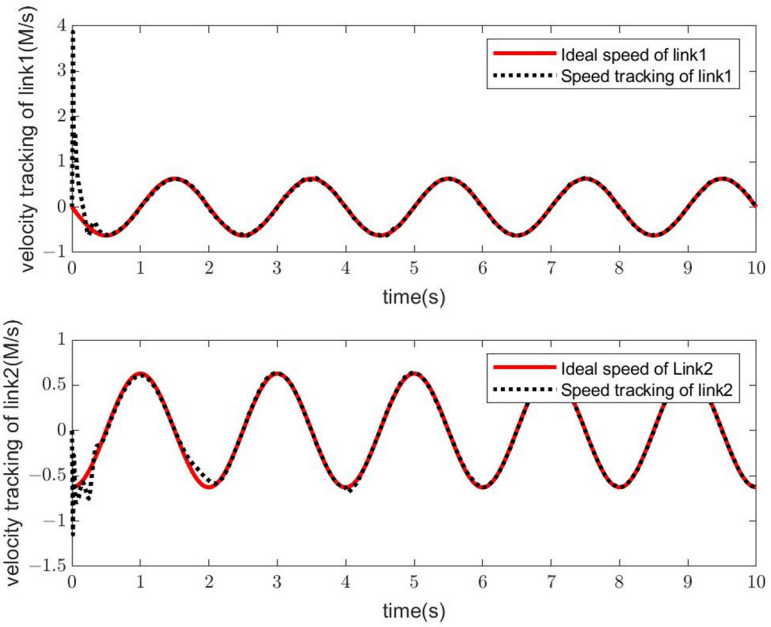

Figure 3. Approximation of the angular velocity link 1 and link 2 in the case of adaptive Elman compensation.

The parameters of all Gaussian functions are taken as $c_i = \begin{bmatrix} -1.5 & -1.0 & -0.5 & 0 & 0.5 & 1.0 & 1.5 \end{bmatrix}$ and $b_i = 10$, The initial threshold of the neural network is set to zero in the simulation. The control law adopts Formula (26), and the adaptive law adopts Formula (31). The benefit is chosen as $K = \begin{bmatrix} 30 & 0 \\ 0 & 30 \end{bmatrix}, k_s = 0.5$. From Lemma 2, it is desirable that $\Lambda = \begin{bmatrix} 15 & 0 \\ 0 & 15 \end{bmatrix}$. The parameters of the adaptive law (31) are taken as $\backslash\Gamma_k = \text{diag}\{2.0\}, Q_k = \text{diag}\{0.10\}$ and $N_k = \text{diag}\{5.0\}$. The simulation results are shown in Figures 5 and 6.

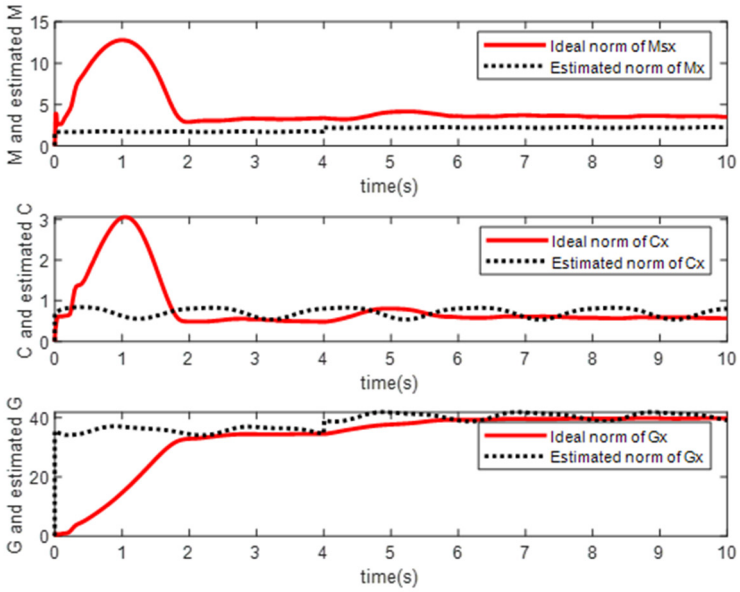

Figure 4. Approximation for the variables $\| M_x(q) \|$, $\| C_x(q, \dot{q}) \|$ and $\| G_x(q) \|$ by Elman neural network.

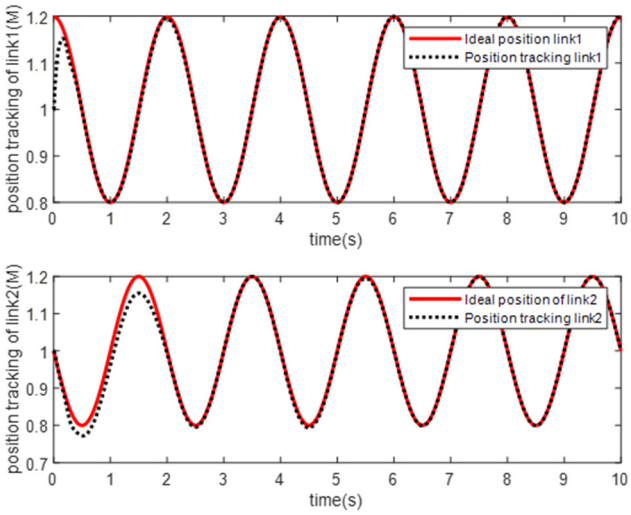

Figure 5. Approximation of the positions of link 1 and link 2 in the case of adaptive RBF compensation.

As can be seen from the Figures 5 and 6, at the beginning of the simulation, the error value was relatively large due to the neural network learning phase of the control input. When the neural network compensator passed the learning stage, the errors were basically eliminated, and the motion trajectory and the estimated value converged in the RBF neural network relatively quickly, but the neural network training and compensation accuracy were lower than Elman neural network. The unknown perturbation was almost completely cancelled after 1 s.

From Figure 7 can be seen that since the tracking trajectory was not a continuous excitation, the estimated values $\| \widehat{M}_x(q) \|$, $\| \widehat{C}_x(q, \dot{q}) \|$, and $\| \widehat{G}_x(q) \|$ did not converge to $\| M_x(q) \|$, $\| C_x(q, \dot{q}) \|$, and $\| G_x(q) \|$, which is often encountered in practical engineering.

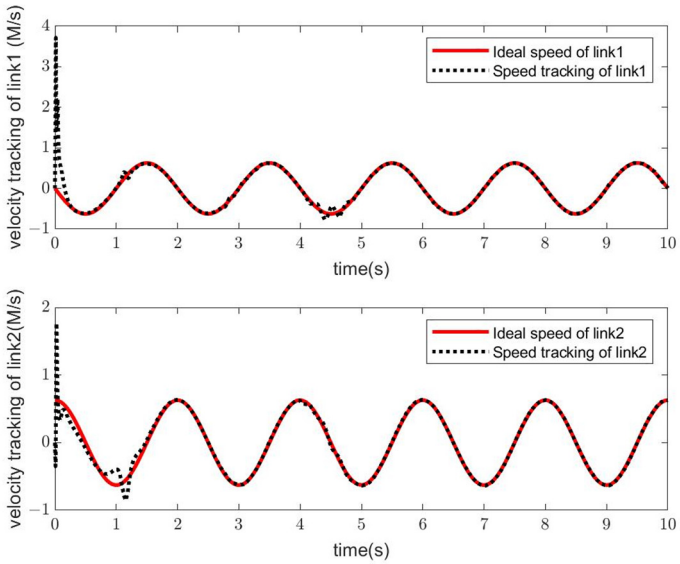

Figure 6. Approximation of the angular velocity link 1 and link 2 in the case of adaptive RBF compensation.

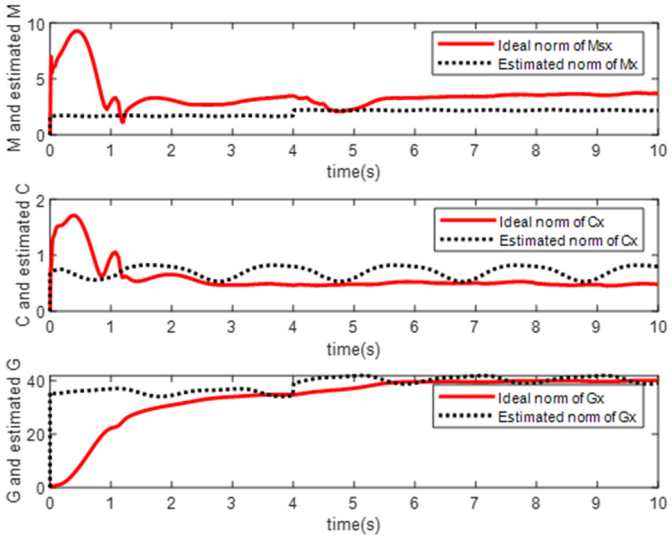

Figure 7. Approximation for the variables $\| M_x(q) \|$, $\| C_x(q, \dot{q}) \|$ and $\| G_x(q) \|$ by RBF adaptive neural network.

From the simulation of the two neural networks, it can be seen from Figures 8 and 9 that although the RBF neural network has faster training and learning speed under the same trajectory planning. But Elman neural network is better in terms of compensation accuracy for external disturbances and errors. However, when there were more interference conditions and uncertain environments, the overall training results of the RBF were better.

In theory, as enough neurons are added to the hidden layer of the Elman network, higher learning expectations will be obtained. However, in the actual simulation, once the number of neurons in the hidden layer exceeds 60, the neural network will be in a state of over-learning and the accuracy will drop.

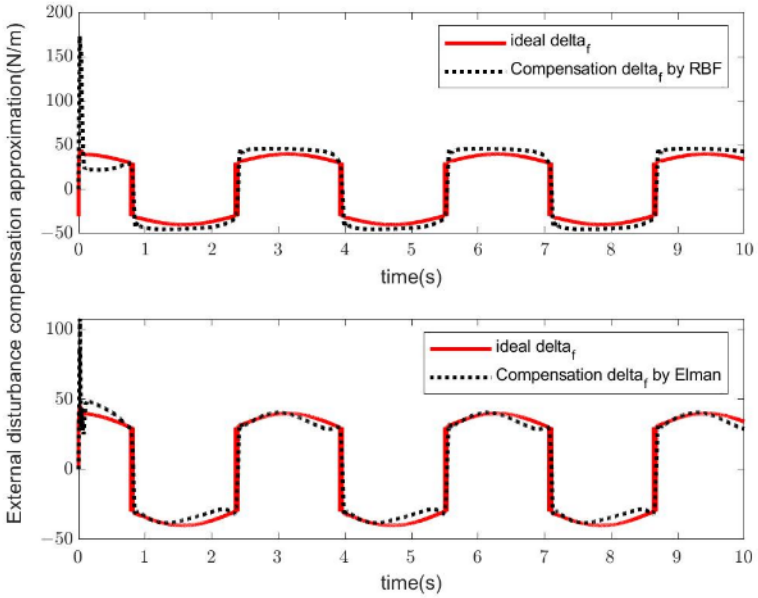

Figure 8. External Disturbance Compensation by Elman Networks and RBF Networks.

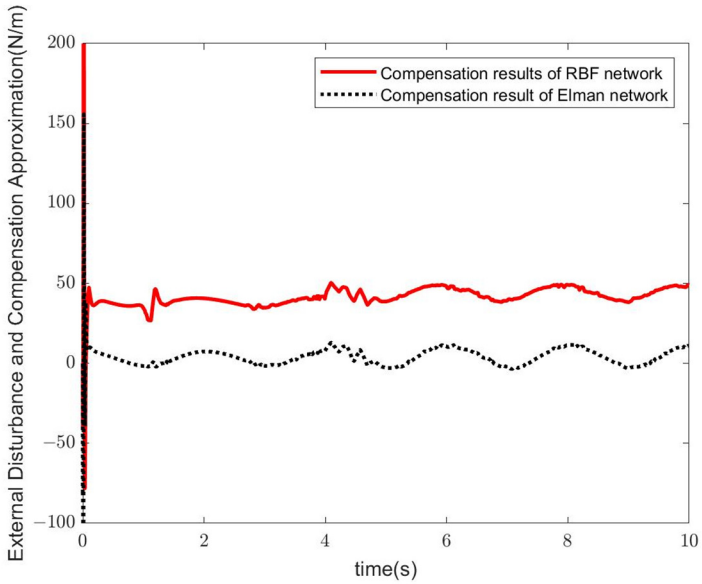

Figure 9. Comparative analysis of external disturbance compensation of the Elman network and RBF network.

5. Conclusions

Therefore, summarizing the research results, the following conclusions can be drawn: During the research process, the quality of a robot to perform a specific task depends not only on the quality of the manufacturing materials and the quality of the moving parts, but also on the quality of the mathematical model of the robot. The control efficiency and accuracy of the robot is based on the analysis of its dynamic model, reducing the error between the planned trajectory and the actual trajectory. This paper presents two different neural networks for manipulating nonlinear dynamic systems of robots by developing models of adaptive neural control schemes based on Elman networks. The choice of network architecture is sound. Simulations of the RBF adaptive neural network show that, despite the faster response time, in the absence of continuous excitation, the system does not converge and the error value is much larger than that of the Elman network. However,

if the simulation is performed locally, or if the system dynamics structure is complex and there are many external disturbances, the response time and accuracy of the RBF adaptive neural network will be better. The rotation model of the planar manipulator is described by the dynamic model equation and the output equation. The simulation and training of an adaptive neural network controller confirms its use in the presence of imprecise dynamic structural data and unknown external disturbances. A computer simulation was applied to the optimal control model, tracking the rotation angle of the manipulator, which confirmed the theoretical position and demonstrated its high operating efficiency. This method can also be applied to the simulation of multi-degree-of-freedom manipulators. At the same time, the improved model can be used for the adaptive control method of the manipulator.

Author Contributions: Conceptualization, methodology, visualization, software, validation, formal analysis, Z.Y., Y.K.; data curation, Z.Y.; writing—original draft preparation, Z.Y., Y.K. and L.X.; writing—review and editing, Z.Y., Y.K. and L.X. All authors have read and agreed to the published version of the manuscript.

Funding: The strategic academic leadership program 'Priority 2030' (Agreement 075-15-2021-1333 dated 30 September 2021).

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Numerical data and simulations are available for academic purposes upon request by contacting the author.

Acknowledgments: The authors would like to thank the Peter the Great St. Petersburg Polytechnic University (SPBSUTU) for providing all facilities during this study.

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

References

1. Duka, A.V. Neural Network based Inverse Kinematics Solution for Trajectory Tracking of a Robotic Arm. *Procedia Technol.* **2014**, *12*, 20–27. [CrossRef]
2. Arseniev, D.G.; Overmeyer, L.; Kälviäinen, H.; Katalinić, B. *Cyber-Physical Systems and Control*; Springer: Berlin/Heidelberg, Germany, 2019.
3. Islam, S.; Liu, X.P. Robust Sliding Mode Control for Robot Manipulators. *IEEE Trans. Ind. Electron.* **2011**, *58*, 2444–2453. [CrossRef]
4. Yazdanpanah, M.J.; KarimianKhosrowshahi, G. Robust Control of Mobile Robots Using the Computed Torque Plus H∞ Compensation Method. Available online: https://www.sciencegate.app/document/10.1109/cdc.2003.1273069 (accessed on 29 June 2022).
5. Rostova, E.N.; Rostov, N.V.; Yan, Y.Z. Neural network compensation of dynamic errors in a position control system of a robot manipulator. *Comput. Telecommun. Control.* **2020**, *64*, 53–64. [CrossRef]
6. Yesildirak, A.; Lewis, F.W.; Yesildirak, S.J. *Neural Network Control of Robot Manipulators and Non-Linear Systems*; CRC Press: Boca Raton, FL, USA, 2020.
7. Kara, K.; Missoum, T.E.; Hemsas, K.E.; Hadjili, M.L. Control of a robotic manipulator using neural network based predictive control. 2010. In Proceedings of the 17th IEEE International Conference on Electronics, Circuits and Systems, Athens, Greece, 12–15 December 2010; pp. 1104–1107. [CrossRef]
8. Seshagiri, S.; Khalil, H. Output Feedback Control of Nonlinear Systems Using RBF Neural Networks. *IEEE Trans. Neural Netw.* **2000**, *11*, 69–79. [CrossRef] [PubMed]
9. Tetko, V.I.; Kůrková, V.; Karpov, P.; Theis, F. *Artificial Neural Networks and Machine Learning–ICANN 2019: Deep Learning: 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17–19, 2019, Proceedings, Part II*; Springer: Berlin/Heidelberg, Germany, 2019.
10. Dou, M.; Qin, C.; Li, G.; Wang, C. Research on Calculation Method of Free flow Discharge Based on Artificial Neural Network and Regression Analysis. *Flow Meas. Instrum.* **2020**, *72*, 101707. [CrossRef]
11. Ren, G.; Cao, Y.; Wen, S.; Huang, T.; Zeng, Z. A Modified Elman Neural Network with a New Learning Rate Scheme. *Neurocomputing* **2018**, *286*, 11–18. [CrossRef]
12. Design and Implementation of a RoBO-2L MATLAB Toolbox for a Motion Control of a Robotic Manipulator. Available online: https://ieeexplore.ieee.org/document/7473678/ (accessed on 30 June 2022).
13. Cheng, Y.-C.; Qi, W.-M.; Cai, W.-Y. Dynamic properties of Elman and modified Elman neural network. In Proceedings of the International Conference on Machine Learning and Cybernetics, Beijing, China, 4–5 November 2002; pp. 637–640. [CrossRef]
14. Beheim, L.; Zitouni, A.; Belloir, F. New RBF neural network classifier with optimized hidden neurons number. *WSEAS Trans. Syst.* **2004**, *2*, 467–472.

15. Song, Q.; Meng, G.J.; Yang, L.; Du, D.Q.; Mao, X.F. Comparison between BP and RBF Neural Network Pattern Recognition Process Applied in the Droplet Analyzer. *Appl. Mech. Mater.* **2014**, *543–547*, 2333–2336. [CrossRef]
16. Luo, B.; Liu, D.; Yang, X.; Ma, H. H ∞ Control Synthesis for Linear Parabolic PDE Systems with Model-Free Policy Iteration. In *Advances in Neural Networks—ISNN 2015*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 81–90. [CrossRef]
17. Ge, S.S.; Hang, C.C.; Woon, L.C. Adaptive neural network control of robot manipulators in task space. *IEEE Trans. Ind. Electron.* **1997**, *44*, 746–752. [CrossRef]
18. Chen, C.; Liu, Z.; Xie, K.; Zhang, Y.; Chen, C.L.P. Adaptive neural control of MIMO stochastic systems with unknown high-frequency gains. *Inf. Sci.* **2017**, *418*, 513–530. [CrossRef]
19. Chen, Y.; Liu, J.; Wang, H.; Pan, Z.; Han, S. Model-free based adaptive RBF neural network control for a rehabilitation exoskeleton. In Proceedings of the 2019 Chinese Control and Decision Conference (CCDC), Nanchang, China, 3–5 June 2019; pp. 4208–4213. [CrossRef]
20. Wang, M.; Yang, A. Dynamic Learning from Adaptive Neural Control of Robot Manipulators with Prescribed Performance. *IEEE Trans. Syst. Man Cybern. Syst.* **2017**, *47*, 2244–2255. [CrossRef]
21. Tran, M.-D.; Kang, H.-J. Nonsingular Terminal Sliding Mode Control of Uncertain Second-Order Nonlinear Systems. *Math. Probl. Eng.* **2015**, *2015*, e181737. [CrossRef]
22. Ortega, R.; Spong, M.W. Adaptive motion control of rigid robots: A tutorial. In Proceedings of the 27th IEEE Conference on Decision and Control, Austin, TX, USA, 7–9 December 1988; pp. 1575–1584. [CrossRef]
23. Zabikhifar, S.; Markasi, A.; Yuschenko, A. Two link manipulator control using fuzzy sliding mode approach. *Her. Bauman Mosc. State Tech. Univ. Ser. Instrum. Eng.* **2015**, *6*, 30–45. [CrossRef]