KSME

Springer

Journal of Mechanical Science and Technology 36 (3) 2022

Original Article
DOI 10.1007/s12206-022-0234-3

Keywords:
· Adaptive neural network
· Robotic manipulator
· EPSO
· Position and velocity tracking
· Task space

Correspondence to:
Zhenbang Xu
xuzhenbang@ciomp.ac.cn

Citation:
Sai, H., Xu, Z., Xu, C., Wang, X., Wang, K., Zhu, L. (2022). Adaptive local approximation neural network control based on extraordinariness particle swarm optimization for robotic manipulators. Journal of Mechanical Science and Technology 36 (3) (2022) 1469~1483. http://doi.org/10.1007/s12206-022-0234-3

Received January 29th, 2021
Revised September 23rd, 2021
Accepted November 14th, 2021

† Recommended by Editor Ja Choon Koo

© The Korean Society of Mechanical Engineers and Springer-Verlag GmbH Germany, part of Springer Nature 2022

Adaptive local approximation neural network control based on extraordinariness particle swarm optimization for robotic manipulators

Huayang Sai[1,2], Zhenbang Xu[1,3], Ce Xu[1,2], Xiaoming Wang[1,2], Kai Wang[1,2] and Lin Zhu[1,2]

[1]Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Science, Jilin Changchun 130033, China, [2]University of Chinese Academy of Sciences, Beijing 100049, China, [3]The Center of Materials Science and Optoelectronics Engineering, University of Chinese Academy of Sciences, Beijing 100049, China

Abstract In this paper, an adaptive radial basis function neural network (RBFNN) controller based on extraordinariness particle swarm optimization (EPSO) is proposed. To improve the trajectory tracking performance of robotic manipulators, the uncertainties of the manipulator dynamic equation are locally approximated using three RBFNNs with optimized hyperparameters. Besides, a robust control item is also considered in the controller to resist external disturbances. During hyperparameters optimization, the EPSO optimizer iteratively optimizes the hyperparameters of the RBFNN controller using the composite error of the system output. The stability of the control scheme is analyzed with the Lyapunov stability. Simulation results as well as the experimental verification prove the efficiency and applicability of the control scheme.

1. Introduction

Nowadays, robotic manipulators are used to assist humans and work in many specific fields such as military, medical science, aerospace, etc., mostly requiring high precision [1]. However, it is difficult to establish an accurate dynamic model of the manipulator because of uncertainties and external disturbances, such as payload changes, friction, external disturbances and sensor noise [2]. To address the problem of inaccurate modeling parameters in the manipulator modeling process and ensure the high tracking accuracy of manipulators, many scholars and research institutions have proposed a variety of methods [3-5]. Among them, adaptive nonlinear control for radial basis function neural network (RBFNN) approximation has been widely studied in recent years.

Narendra et al. [6] first formulated an artificial neural network adaptive control method for nonlinear dynamic systems, which laid the foundation for the application of RBFNNs in robot control. To address the complex problem of the inverse solution of the Jacobian matrix in the robot adaptive control system, a neural network online modeling method was proposed in Ref. [7]. Subsequently, multilayer neural networks and RBFNNs have been successfully applied in the field of robot control [8, 9]. For instance, to overcome the problem of the explosion of the flexible joint manipulator dynamic model in backstepping control, the control method of model-free compensators using a neural network and sliding surface was proposed, which constructs an adaptive RBFNN to approximate the unknown function of the system model, and uses the nonlinear damping term to overcome the external disturbance torque [10, 11]. Although RBFNNs are widely used, how to improve the control effect and anti-disturbance ability of the neural network control method is still a problem worth further study.

In recent years, researchers have attempted to improve the performance of robot controllers and optimize the mechanical structure by introducing different structures of neural networks [12-14]. Neural network structures, such as an echo state network and convolutional neural

network, have also been successfully applied in the robot control field [15, 16]. Azizi [17] proposed an artificial neural network architecture to find the optimal parameters of the universal joint, which helps to reduce energy consumption and waste in the manufacturing process. In Ref. [18], a ring probabilistic logic neural network architecture is designed to reduce the adverse effects of road uncertainty on vehicles, and it is beneficial to improve the service life of vehicles. Notably, most neural network structures have many hyperparameters that need to be determined, and the pros and cons of the hyperparameters have a great influence on the control effect of robots. Some scholars have proposed schemes to select the hyperparameters of neural networks [19-21]. For example, for the RBFNN, the classical approach used to locate neuron centers is to apply clustering techniques, such as k-means clustering [22] and vector quantization [23], to form templates of the input. However, current conventional clustering algorithms, such as the k-means algorithm, cannot ensure the high precision of the selected centers. Ismkhan [19] and Hu et al. [24] proposed an improved k-means algorithm, which yields good accuracy but has a large computational cost. Additionally, the gradient descent algorithm, orthogonal least squares and support vector machine are also used to determine hyperparameters in neural networks [25]. These methods are proved that hyperparameter values have a great impact on the performance of neural networks. However, these methods require a large amount of data sets and computation, which is difficult to apply in the field of robot control. In generally, the hyperparameters of neural networks are mostly artificially selected to be used in robot control systems, which is time and effort-consuming and in great random.

Inspired by physical phenomena, the metaheuristic optimization algorithm provides a tool for researchers to optimize target parameters. Most of these algorithms are developed by drawing inspiration from the natural world and are very suitable for solving complex calculation problems in design and operation optimization problems [26-28]. Particle swarm optimization (PSO) is one of the most famous metaheuristic optimization algorithms, and mainly imitates the foraging behavior of birds. However, in the standard PSO algorithm, particles tend to fall into a local extremum, and premature convergence or stagnation occurs. To address this problem, scholars have proposed many improved algorithms based on the PSO algorithm [29-31]. Among them, Ngo et al. [31] proposed the extraordinariness particle swarm optimization (EPSO) algorithm, which overcomes the shortcomings of the standard PSO algorithm, by moving to other selected particles and altering the range of the potential target. EPSO algorithm enhances the performance of standard PSO by using a new movement strategy for each particle. Specifically, the particles in EPSO fly towards their own predetermined goals, rather than the best particles (i.e., personal and global bests). However, the EPSO has only been simply applied in the design of mechanical structure at present, and its potential in manipulator control has not been further explored.

In this paper, the effects of hyperparameters on robot control performance when using RBFNN to approximate the robot dynamic model is investigated. We propose an adaptive RBFNN control scheme based on an EPSO optimizer (EPRBF) for robotic systems with external disturbances in task space, which includes both an inner-loop and outer-loop control structure. The inner-loop controller is designed to model the manipulator using RBFNN based on local approximation. The outer loop controller is designed to use a proportion-derivative (PD) controller, and a robust controller is designed to quickly eliminate position and speed tracking errors. Meanwhile, the EPSO optimizer is introduced to optimize the hyperparameters in the RBFNN controller using the composite error to eliminate the errors caused by artificially selecting hyperparameters. Finally, numerical simulation and experimental results are shown to verify the effectiveness of and the advantages of the designed scheme by comparing it with some existing ones.

The structure of this paper is as follows: In Sec. 2, we discuss theoretical preparatory knowledge, including details of the mathematical notation, EPSO, RBFNN, and robotic system dynamic model. In Sec. 3, we design the EPRBF controller and discuss stability. We present the simulation results of the proposed scheme in Sec. 4 and experiments that are conducted in Sec. 5, followed by conclusions in Sec. 6.

2. Problem formulation and preliminaries

In this paper, R, R^n and $R^{n\times n}$ denote the real number set, n-dimensional vector space and n×n real matrix space, respectively. L_2 denotes the Euclidean norm for vectors, and L_∞ denotes the largest absolute value of the element in x vector space. $\hat{\alpha}$ is an estimate of arbitrary parameter α, which specifies that $\tilde{\alpha}=\alpha-\hat{\alpha}$, and $\tilde{\alpha}$ denotes the error value between the actual value and estimate of α.

2.1 Description of EPSO

For the EPSO algorithm, a combined operator is utilized instead of the coefficients presenting cognitive and social components in the standard PSO. That is to say, any particles in the EPSO algorithm can exchange information with each other, which can help particles escape from the local optimum, as particles do not always move towards their best locations. The detailed procedure of the EPSO algorithm is as follows [31].

First, initialize the particle population, i.e., randomly generate some new particles between the given upper and lower bounds. Then, these particles are sorted from the best to the worst according to their cost/fitness. Next, the target index T needs to be selected for each particle at each iteration:

$$T=round(rand\times N_{pop}) \tag{1}$$

where N_{pop} is the population size and $rand \in [0, 1]$ is a uniformly distributed random number. Then, the particle position update formula is as follows:

$$\vec{X}_i(t+1)=\begin{cases}\vec{X}_i(t)+C_{epso}\left(\vec{X}_{Ti}(t)-\vec{X}_i(t)\right)\\ \text{if } T\in(0,\ T_{up})\\ L\vec{B}_i+rand\times\left(U\vec{B}_i-L\vec{B}_i\right)\\ \text{otherwise}\end{cases} \tag{2}$$

where *LB* and *UB* are the lower and upper bounds of the search space, respectively. $C_{epso}\in$ [0, 2] denotes a combined component that includes cognitive and social factors. Finally, the stop condition is checked until the particle satisfies the stop criterion.

According to the above description, the main feature of the EPSO algorithm is to change the range of potential targets and make them move to other selected particles, which brings the opportunity for particles to jump out of the local optimum, to overcome the shortcomings of premature convergence of PSO algorithm.

2.2 Description of the RBFNN

The RBFNN can approximate any nonlinear function in a compact set with arbitrary precision [32, 33]. As shown in Fig. 1, the RBFNN has a three-layer network structure, which consists of an input layer, hidden layer and output layer. Xp represents the sample space and $x=[x_1,\cdots,\ x_M]^T\in R^M$ is the input vector. In the RBFNN, the neuron activation function of the hidden layer is often composed of the RBF. The Gaussian function is the most commonly used RBF because of its simple expression, good analyticity and smoothness. An array operation unit composed of hidden layers is called a hidden layer node. Each hidden layer node contains a center vector c with the same dimension as the input parameter vector x, and the Euclidean distance between them is defined as $\|x(t)-c_j(t)\|$.

The output of the hidden layer is composed of a nonlinear activation function $h_j(t)$:

$$h_j(t)=\exp\left(-\frac{\|x(t)-c_j(t)\|}{2b_j^2}\right),\ j=1,\cdots,\ N,\quad t=1,\cdots,\ M, \tag{3}$$

where b_j is a positive scalar that represents the width of the Gaussian function. N is the number of nodes in the hidden layer. The output layer of the network is implemented by the following weighting function:

$$y_i(t)=\sum_{j=1}^{N}w_{ji}h_j(t),\ i=1,\cdots,\ J,\ \ t=1,\cdots,\ M, \tag{4}$$

where w is the weight of the output layer, J is the number of nodes in the output layer and y is the output of the neural network.

At present, some results [34-36] indicate that for any continuous smooth function $f(x)$: Θ_x over a compact set $\Theta_x\in R^M$, if RBFNN is applied to approximate the nonlinear function $f(x)$ and the number of hidden layer nodes Φ_N is sufficiently large, then a set of ideal bounded weights w^* exist, and we have

$$f(x)=w^{*T}h(x)+\varepsilon(x) \tag{5}$$

where $\varepsilon(x)$ is the reconstruction error. Note that the ideal weight matrix w^* in Eq. (5) is assumed to be completely known. However, in an actual application process, the RBFNN is typically constructed using the estimated value $\hat{w}$ of the weight instead of the ideal weight w^* to approximate the unknown function $f(x)$, that is

$$\hat{f}(x)=\hat{w}^Th(x) \tag{6}$$

where $\hat{w}$ is the estimated weight matrix, which can be adjusted using a weight learning law.

2.3 Robot dynamics modeling

In this paper, the dynamic can be obtained using the Lagrange equation for an *n*-DOF rigid manipulator with bounded random disturbances

$$M(q)\ddot{q}+C(q,\dot{q})\dot{q}+G(q)=\tau-\tau_d \tag{7}$$

where q, $\dot{q}$ and $\ddot{q}\in R^n$ are the joint angular position vectors, velocity vectors and acceleration vectors of the manipulator, respectively; $M(q)\in R^{n\times n}$ is the symmetric and positive definite inertia matrix; $C(q,\dot{q})\in R^{n\times n}$ is the centrifugal and Coriolis forces matrix; $G(q)\in R^n$ is the gravity vector; $\tau\in R^n$ is the torque input vector, and $\tau_d\in R^n$ represents the unknown disturbances.

In practical engineering applications, manipulators are affected by external disturbances and system uncertainties. Uncertainty mainly refers to the factors that are not considered or

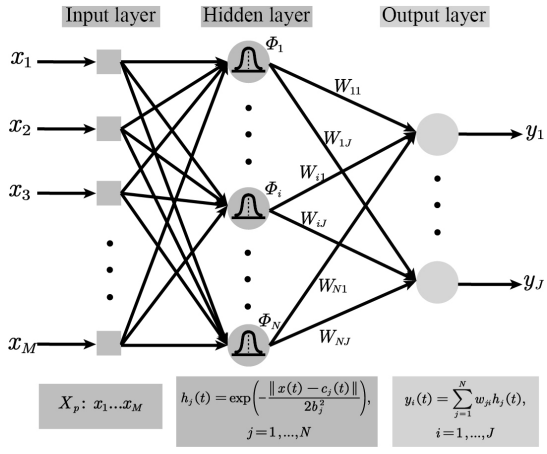


Fig. 1. Structure of the RBFNN.

are intentionally ignored when establishing the mathematical model of the manipulator [37], including inaccuracies in the quality and length of the manipulator, inaccuracies in the parameters of the gravitational acceleration and the influence of the frictional force when the joint is rotated. Generally, $M(q)$, $C(q,\dot{q})$ and $G(q)$ are unknown, so we can assume that $M_0(q)$, $C_0(q,\dot{q})$ and $G_0(q)$ are the nominal parts of the model parameters, and the errors are denoted by E_M, E_C and E_G, respectively. These variables can be expressed as

$$\begin{cases} M(q)=M_0(q)+E_M \\ C(q,\dot{q})=C_0(q,\dot{q})+E_C \\ G(q)=G_0(q)+E_G \end{cases} \tag{8}$$

where E_M, E_C and E_G are the modeling errors of $M(q)$, $C(q,\dot{q})$ and $G(q)$, respectively.

Assuming that the working nature of the manipulator is related to the spatial position of the end effector, it is necessary to design the control algorithm directly in the workspace. Using x to indicate the position and orientation of the end effector in the workspace, the dynamics of the robot in the workspace can be expressed as [38]

$$M_x(q)\ddot{x}+C_x(q,\dot{q})\dot{x}+G_x(q)=F_x, \tag{9}$$

where $M_x(q)=J^{-T}(q)M(q)J^{-1}(q)$; $C_x(q,\dot{q})=J^{-T}(q)\left(C(q,\dot{q})-D(q)J^{-1}(q)\dot{J}(q)\right)J^{-1}(q)$; $G_x(q)=J^{-T}(q)G(q)$ and $F_x=J^{-T}(q)\tau$; $J(q)\in R^{n\times n}$ are Jacobian matrices determined by the manipulator system, and we assume that they are nonsingular in the bounded workspace. Simultaneously, the dynamic equation of the manipulator has the following characteristics.

Property 1. The inertia matrix M is a symmetric and positive definite inertia matrix.

Property 2. If $C_x(q,\dot{q})$ is defined by the Christoffel symbol rule, then the matrix $\dot{M}_x(q)$ - $2C_x(q,\dot{q})$ is diagonally symmetric.

Property 3. Unknown disturbance τ_d is bounded and satisfies

$$\|\tau_d\|\leq\tau_D \tag{10}$$

where τ_D is a positive constant.

3. Controller design and stability analysis

In this section, to improve the position tracking performance, and anti-interference ability of manipulators, the EPRBF controller is proposed and its stability is analyzed.

3.1 Neural network modeling of the manipulator

As discussed in Sec. 2, $M(q)$, $C(q,\dot{q})$ and $G(q)$ are often unknown in actual engineering applications. We use three RBFNNs to approximate $M(q)$, $C(q,\dot{q})$ and $G(q)$. From Eq. (8): $M(q)$ and $G(q)$ are functions about q, so they can be modeled using a static neural network. Because $C(q,\dot{q})$ is a function of q and $\dot{q}$, it needs to be modeled using a dynamic neural network with inputs q and $\dot{q}$. The three output items of the RBFNN are $M_{SNN}(q)$, $C_{DNN}(q,\dot{q})$ and $G_{SNN}(q)$, which can be expressed as

$$\begin{cases} M(q)=M_{SNN}(q)+E_M \\ C(q,\dot{q})=C_{DNN}(q,\dot{q})+E_C \\ G(q)=G_{SNN}(q)+E_G \end{cases} \tag{11}$$

where E_M, E_C and E_G are the approximation errors of $M(q)$, $C(q,\dot{q})$ and $G(q)$, respectively.

Similarly, in Eqs. (7) and (9), $M_x(q)$, $C_x(q,\dot{q})$, $G_x(q)$ and $M(q)$, $C(q,\dot{q})$, $G(q)$ have similar properties. Therefore, as shown in Fig. 2, models of neural networks can be built as follows:

$$\begin{cases} m_{xkj}(q)=\sum_l \theta_{kjl}\xi_{kjl}(q)+\varepsilon_{mkj}(q) \\ \quad\quad =\theta_{kj}^T\xi_{kj}(q)+\varepsilon_{mkj}(q) \\ g_{xk}(q)=\sum_l \beta_{kl}\eta_{kl}(q)+\varepsilon_{gk}(q) \\ \quad\quad =\beta_k^T\eta_k(q)+\varepsilon_{gk}(q) \\ c_{xkj}(q,\dot{q})=\sum_l \alpha_{kjl}\zeta_{kjl}(z)+\varepsilon_{ckj}(z) \\ \quad\quad =\alpha_{kj}^T\zeta_{kj}(z)+\varepsilon_{ckj}(z) \end{cases} \tag{12}$$

where $m_{xkj}(q)$, $g_{xk}(q)$, $c_{xkj}(q,\dot{q})$ are the three output items of the RBFNNs; θ_{kjl}, β_{kl}, $\alpha_{kjl}\in R$ are the weights of the RBFNNs; $\xi_{kjl}(q)\in R$, $\eta_{kl}(q)\in R$ are the corresponding Gaussian basis functions with input vector q; $z=[q^T\ \dot{q}^T]^T\in R^{2n}$; $\zeta_{kjl}(z)\in R$ is the corresponding Gaussian basis function with input vector z; and $\varepsilon_{mkj}(q)$, $\varepsilon_{gk}(q)\in R$, $\varepsilon_{ckj}(z)\in R^{2n}$ are the modeling errors of $m_{xkj}(q)$, $g_{xk}(q)$ and $c_{xkj}(q,\dot{q})$, and we

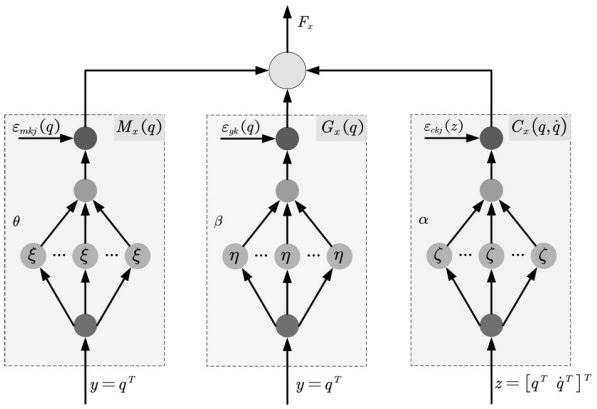


Fig. 2. Local approximation of $M_x(q)$, $C_x(q,\dot{q})$, $G_x(q,\dot{q})$ using RBFNNs.

assume that they are all bounded.

When building the robotic dynamic model with neural networks, the dynamic equation of the manipulator in the workspace can be expressed as

$$M_x(q)\ddot{x}+C_x(q,\dot{q})\dot{x}+G_x(q)=F_x \tag{13}$$

where $M_x(q)$, $C_x(q,\dot{q})$, $G_x(q)$ can be expressed as

$$\begin{cases} m_{xkj}(q)=\theta_{kj}^T\xi_{kj}(q)+\varepsilon_{dkj}(q) \\ c_{xkj}(q,\dot{q})=\alpha_{kj}^T\zeta_{kj}(z)+\varepsilon_{ckj}(z). \\ g_{xk}(q)=\beta_k^T\eta_k(q)+\varepsilon_{gk}(q) \end{cases} \tag{14}$$

Using the General-Leeway (GL) matrix and its multiplication operation [38], $M_x(q)$ can be written as

$$M_x(q)=[\{\Theta\}^T\cdot\{Q(q)\}]+E_M(q) \tag{15}$$

where $\{\Theta\}$ and $\{Q(q)\}$ are GL matrices whose elements are θ_{kj} and $\xi_{kj}(q)$, respectively; and $E_M(q)\in R^{n\times n}$ is the matrix whose elements are the modeling errors.

Simultaneously, for $C_x(q,\dot{q})$ and $C_x(q,\dot{q})$, we have

$$C_x(q,\dot{q})=[\{A\}^T\cdot\{Z(z)\}]+E_c(z), \tag{16}$$

$$G_x(q)=[\{B\}^T\cdot\{H(q)\}]+E_G(q) \tag{17}$$

where $\{A\}$, $\{Z(z)\}$, $\{B\}$ and $\{H(q)\}$ are GL matrices and GL vectors whose elements are α_{kj}, $\zeta_{kj}(z)$, β_k and $\eta_k(q)$, respectively; and $E_c(z)\in R^{n\times n}$ and $E_G(q)\in R^n$ are the matrix and vector whose elements are the modeling errors $\varepsilon_{ckj}(z)$ and $\varepsilon_{gk}(q)$.

3.2 Scheme design based on the EPRBF controller

We define that $x_d(t)$ as the ideal trajectory of the manipulator in the workspace. Then $\dot{x}_d(t)$ and $\ddot{x}_d(t)$ are their ideal velocities and accelerations, respectively. The manipulator position tracking error $e(t)$, neural network modeling speed $\dot{x}_r(t)$ and composite error $r(t)$ are as follows:

$$\begin{cases} e(t)=x_d(t)-x(t) \\ \dot{x}_r(t)=\dot{x}_d(t)+\Lambda e(t) \\ r(t)=\dot{e}(t)+\Lambda e(t) \end{cases} \tag{18}$$

where Λ is the ratio of the position error of the manipulator to the velocity error, and Λ is a positive definite matrix. The composite error $r(t)$ consists of two parts: the speed error and position error of the robotic manipulator.

In Fig. 3, the EPRBF controller is divided into two parts: an inner-loop controller and outer-loop controller. The inner-loop controller mainly includes three adaptive RBFNNs and a manipulator dynamic model for approximating $M(q)$, $C(q,\dot{q})$, $G(q)$. The outer-loop controller is designed to use a PD controller and a robust controller to quickly eliminate position and speed tracking errors. Additionally, the outer-loop controller includes an EPSO optimizer, which is used to optimize the hyperparameters in the RBF controller using the composite error to eliminate errors caused by artificially selecting hyperparameters. In the formulation of the controller of robotic manipulators with uncertain models, the hyperparameters in the neural network are used as the optimization objective of the EPSO optimizer, and the trajectory tracking error of the manipulator is taken as the optimization target.

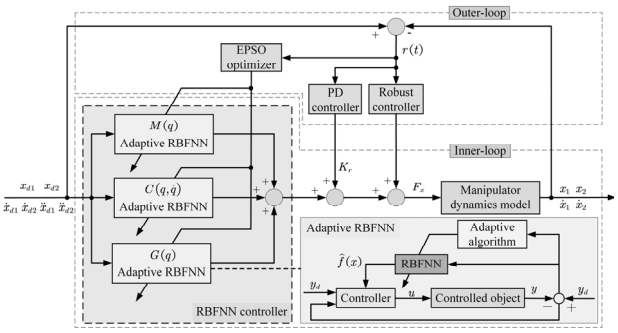


Fig. 3. The control block diagram of the EPRBF controller.

According to the provisions in Sec. 2, $\{\hat{A}\}$,$\{\hat{B}\}$ and $\{\hat{\Theta}\}$ are estimates of $\{\hat{A}\}$,$\{\hat{B}\}$ and $\{\Theta\}$, respectively. The control law is designed as follows:

$$\begin{aligned} F_x=&\left[\{\hat{\Theta}\}^T\cdot\{Q(q)\}\right]\ddot{x}_r+\left[\{\hat{A}\}^T\cdot\{Z(z)\}\right]\dot{x}_r \\ &+\left[\{\hat{B}\}^T\cdot\{H(q)\}\right]+K_r+K_a\,sat(r) \end{aligned} \tag{19}$$

where $K\in R^{n\times n}>0$ and K_a is a positive constant satisfies $K_a>\|E\|$, $E=E_M(q)\ddot{x}_r+E_c(z)\dot{x}_r+E_G(q)$.

The first three items of the controller are model-based control in Eq. (19). Kr is equivalent to PD control, and r is defined in Eq. (18). The last term of the controller is a robust item against the modeling errors of the neural networks. $\mathrm{sat}(r)$ is a saturation function of r

$$\mathrm{sat}(r)=\begin{cases} 1 & r>\delta \\ r/\delta & |r|\le\delta \\ -1 & r<-\delta \end{cases} \tag{20}$$

where δ is a defined positive constant.

According to Eq. (19), the controller does not require the solution of the Jacobian inverse matrix. In actual control, the input can be obtained using $\tau=J^T(q)F_x$.

The EPSO optimizer adjusts the hyperparameters b and c of each neural network in the RBFNN controller, and the composite error $r(t)$ is used as the objective function of the EPSO

optimizer. In Eq. (3), $b=[b_1,\cdots,\ b_m]^T$, where $b_j>0$ is the width of the Gaussian function of the hidden layer neuron j; $c_j=[c_{j1},\cdots,\ c_{jn}]$ is the center point vector of the *j*th hidden layer neuron. First, the EPSO optimizer randomly generates a set of random particle sets p within a range of values. We design each RBFNN with seven hidden layers; hence, the number of randomly generated random particle sets is 63. *c-M*, *c-G* and *c-C* represent the center point vectors of the hidden layer neurons in $M(q),\ C(q,\dot{q}),\ G(q)$, respectively:

$$\begin{aligned} b&=[p(1)\ p(2)\ p(3)\ p(4)\ p(5)\ p(6)\ p(7)]\\ c-M&=\begin{bmatrix} p(8)\ p(9)\ p(10)\ p(11)\ p(12)\ p(13)\ p(14);\\ p(15)\ p(16)\ p(17)\ p(18)\ p(19)\ p(20)\ p(21)\end{bmatrix}\\ c-G&=\begin{bmatrix} p(22)\ p(23)\ p(24)\ p(25)\ p(26)\ p(27)\ p(28);\\ p(29)\ p(30)\ p(31)\ p(32)\ p(33)\ p(34)\ p(35)\end{bmatrix}\\ c-C&=\begin{bmatrix} p(36)\ p(37)\ p(38)\ p(39)\ p(40)\ p(41)\ p(42);\\ p(43)\ p(44)\ p(45)\ p(46)\ p(47)\ p(48)\ p(49);\\ p(50)\ p(51)\ p(52)\ p(53)\ p(54)\ p(55)\ p(56);\\ p(57)\ p(58)\ p(59)\ p(60)\ p(61)\ p(62)\ p(63)\end{bmatrix}. \end{aligned} \tag{21}$$

According to Eqs. (15)-(17) and the control law in Eq. (19), the tracking error equation can be obtained as follows:

$$\begin{aligned} &M_x(q)\dot{r}+C_x(q,\dot{q})r+Kr+k_a\mathrm{sat}(r)\\ &=\left[\left\{\tilde{\Theta}\right\}^T\cdot\{Q(q)\}\right]\ddot{x}_r+\left[\left\{\tilde{A}\right\}^T\cdot\{Z(z)\}\right]\dot{x}_r\\ &+\left[\left\{\tilde{B}\right\}^T\cdot\{H(q)\}\right]+E. \end{aligned} \tag{22}$$

The detailed derivation process is provided in Appendix.

3.3 Stability analysis

In the RBFNN controller, a neural network weight adaptive strategy is used to adjust the weight of the RBFNN. For the closed-loop system in Eq. (22), the adaptive law is designed as

$$\begin{cases} \dot{\hat{\theta}}_k=\Gamma_k\cdot\{\xi_k(q)\}\ddot{x}_r r_k\\ \dot{\hat{\alpha}}_k=\Psi_k\cdot\{\zeta_k(z)\}\dot{x}_r r_k,\\ \dot{\hat{\beta}}=N_k\eta_k(q)r_k \end{cases} \tag{23}$$

where $\Gamma_k=\Gamma_k^T>0$; $Q_k=Q_k^T>0$; $N_k=N_k^T>0$ and $\hat{\theta}_k$ and $\hat{\alpha}_k$ are vectors of elements $\hat{\theta}_{kj}$ and $\hat{\alpha}_{kj}$, respectively. We can conclude that $\hat{\theta}_k$, $\hat{\alpha}_k$, $\hat{\beta}_k\in L_\infty$, $e\in L_2^n\cap L_2^n$, e is continuous. Hence, $e\to 0$ and $\dot{e}\to 0$ as $t\to\infty$.

Proof. We choose the following Lyapunov function:

$$\begin{aligned} V=&\frac{1}{2}r^TM_x(q)r+\frac{1}{2}\sum_{k=1}^n\tilde{\theta}_k^T\Gamma_k^{-1}\tilde{\theta}_k\\ &+\frac{1}{2}\sum_{k=1}^n\tilde{\alpha}_k^T\Psi_k^{-1}\tilde{\alpha}_k+\frac{1}{2}\sum_{k=1}^n\tilde{\beta}_k^TN_k^{-1}\tilde{\beta}_k \end{aligned} \tag{24}$$

where Γ_k, Ψ_k, N_k are symmetric and positive matrices. Taking the derivative of Eq. (24) yields

$$\begin{aligned} \dot{V}=&r^T\left(M_x(q)\dot{r}+\frac{1}{2}\dot{M}_x(q)r\right)+\sum_{k=1}^n\tilde{\theta}_k^T\Gamma_k^{-1}\dot{\tilde{\theta}}_k\\ &+\sum_{k=1}^n\tilde{\alpha}_k^T\Psi_k^{-1}\dot{\tilde{\alpha}}_k+\sum_{k=1}^n\tilde{\beta}_k^TN_k^{-1}\dot{\tilde{\beta}}_k. \end{aligned} \tag{25}$$

Because the matrix $M_x(q)-2C_x(q,\dot{q})$ is diagonally symmetric, $r^T(M_x-2C_x)r=0$. Therefore, according to Eq. (25), we have

$$\begin{aligned} \dot{V}=&r^T(M_x(q)\dot{r}+C_x(q,\dot{q})r)+\sum_{k=1}^n\tilde{\theta}_k^T\Gamma_k^{-1}\dot{\tilde{\theta}}_k\\ &+\sum_{k=1}^n\tilde{\alpha}_k^T\Psi_k^{-1}\dot{\tilde{\alpha}}_k+\sum_{k=1}^n\tilde{\beta}_k^TN_k^{-1}\dot{\tilde{\beta}}_k. \end{aligned} \tag{26}$$

Then, substituting Eq. (25) into Eq. (26) yields

$$\begin{aligned} \dot{V}=&-r^TKr-k_ar^Tsat(r)+r^TE\\ &+r^T\left[\left\{\tilde{\Theta}\right\}^T\cdot\{Q(q)\}\right]\ddot{x}_r+\sum_{k=1}^n\tilde{\theta}_k^T\Gamma_k^{-1}\dot{\tilde{\theta}}_k\\ &+r^T\left[\left\{\tilde{A}\right\}^T\cdot\{Z(z)\}\right]\dot{x}_r+\sum_{k=1}^n\tilde{\alpha}_k^T\Psi_k^{-1}\dot{\tilde{\alpha}}_k\\ &+r^T\left[\left\{\tilde{B}\right\}^T\cdot\{H(q)\}\right]+\sum_{k=1}^n\tilde{\beta}_k^TN_k^{-1}\dot{\tilde{\beta}}_k. \end{aligned} \tag{27}$$

According to Eq. (25), it has

$$\begin{cases} r^T\left[\left\{\tilde{\Theta}\right\}^T\cdot\{Q(q)\}\right]\ddot{x}_r=\sum_{k=1}^n\left\{\tilde{\theta}_k\right\}^T\{\xi_k(q)\}\ddot{x}_r r_k\\ r^T\left[\left\{\tilde{A}\right\}^T\cdot\{Z(z)\}\right]\dot{x}_r=\sum_{k=1}^n\tilde{\alpha}_k^T\{\zeta_k(z)\}\dot{x}_r r_k\\ r^T\left[\left\{\tilde{B}\right\}^T\cdot\{H(q)\}\right]=\sum_{k=1}^n\tilde{\beta}_k^T\eta_k(q)r_k \end{cases}. \tag{28}$$

Substituting Eq. (28) into Eq. (27) yields

$$\begin{aligned} \dot{V}=&-r^TKr-k_ar^T\mathrm{sat}(r)+r^TE\\ &+\sum_{k=1}^n\left\{\tilde{\theta}_k\right\}^T\cdot\{\xi_k(q)\}\ddot{x}_r r_k+\sum_{k=1}^n\tilde{\theta}_k^T\Gamma_k^{-1}\dot{\tilde{\theta}}_k\\ &+\sum_{k=1}^n\tilde{\alpha}_k^T\cdot\{\zeta_k(z)\}\dot{x}_r r_k+\sum_{k=1}^n\tilde{\alpha}_k^T\Psi_k^{-1}\dot{\tilde{\alpha}}_k\\ &+\sum_{k=1}^n\tilde{\beta}_k^T\eta_k(q)r_k+\sum_{k=1}^n\tilde{\beta}_k^TN_k^{-1}\dot{\tilde{\beta}}_k. \end{aligned} \tag{29}$$

Substituting the adaptive law in Eq. (23) into Eq. (29), combined with inequality $k_a>\|E\|$, $\dot{V}$ is bounded as

$$\dot{V} = -r^T Kr - k_a r^T \mathrm{sat}(r) + r^T E \leqslant -r^T Kr \leqslant 0. \quad (30)$$

As a result of $\dot{V} \leqslant -r^T Kr \leqslant 0$, $\forall t \geqslant 0$, it follows that $0 \leqslant V(t) \leqslant V(0)$. Therefore, as $V(t) \in L_\infty$, this means that r, $\tilde{\theta}_k$, $\tilde{\alpha}_k$ and $\tilde{\beta}_k \in L_\infty$. Note that $r \in L_2^n$, x_d, $\dot{x}_d$, $\ddot{x}_d \in L_\infty^n$ and $\{Q(q)\}$, $\{Z(z)\}$, $\{H(q)\}$ are bounded basis functions, so $\dot{r} \in L_\infty^n$, which means that r is consistent and continuous. When $t \to \infty$ and $r \to 0$, it has $\dot{e} \to 0$ as $t \to \infty$.

4. Numerical example

To prove the effectiveness of the proposed control scheme, a 2-DOF manipulator is used as the simulation object. The architecture of the two-link robotic manipulator is shown in Fig. 4. The EPRBF control scheme is simulated using the fixed force and time-varying force, and compared with the control scheme of the RBFNN controller. Furthermore, the influence of different numbers of iterations on the proposed control scheme is analyzed, and the influences of different PSO improved algorithms on the adaptive RBFNN control scheme of local approximation are compared.

4.1 Design parameter

The analysis of the 2-DOF manipulator model from Sec. 2.3 shows that the dynamic of the robotic manipulator is

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau \quad (31)$$

where

$$M(q) = \begin{bmatrix} m_1 + m_2 + 2m_3 \cos q_2 & m_2 + m_3 \cos q_2 \\ m_2 + m_3 \cos q_2 & m_2 \end{bmatrix}, \quad (32)$$

$$C(q,\dot{q}) = \begin{bmatrix} -m_3 \dot{q}_2 \sin q_2 & -m_3(\dot{q}_1 + \dot{q}_2 \sin q_2) \\ m_3 \dot{q}_1 \sin q_2 & 0 \end{bmatrix}, \quad (33)$$

$$G(q) = \begin{bmatrix} m_4 g \cos q_1 + m_5 g \cos(q_1 + q_2) \\ m_5 g \cos(q_1 + q_2) \end{bmatrix}. \quad (34)$$

The value of m_i can be obtained from $M_m = P + |p_l| L$, where $M_m = [m_1 \ m_2 \ m_3 \ m_4 \ m_5]^T$, $P = [p_1 \ p_2 \ p_3 \ p_4 \ p_5]^T$ and $L = [l_1^2 \ l_2^2 \ l_1 l_2 \ l_1 \ l_2]^T$; p_l is external disturbance; l_1 and l_2 are the length of link 1 and link 2, respectively; and P is the parameter vector of the manipulator.

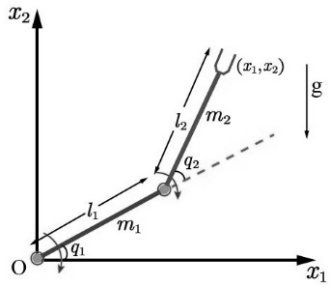


Fig. 4. Architecture of a two-link robotic manipulator.

The nominal parameters of the manipulator are chosen as $P = [1.66 \ 0.42 \ 0.63 \ 3.75 \ 1.25]^T \ \mathrm{kg \cdot m^2}$, $l_1 = l_2 = 1 \ \mathrm{m}$. To compare the ability of the manipulator to resist load disturbances in the two control schemes, two simulation analyses are performed. For the first simulation, the value of p_l at t = 3 s changes from 0 to 2. In the second simulation, the value of the load is a varying force $p_l = \sin(2\pi t)$ at t > 3 s.

The desired position tracking trajectory of end-effector in Cartesian space is $x_{d1}(t) = 1.0 + 0.2\cos(\pi t)$, $x_{d2}(t) = 1.0 + 0.2\sin(\pi t)$. The trajectory is a circle with a radius of 0.2 m and the center of the circle is $(x_1, x_2) = (1.0, 1.0)$. In the initial condition, the end effector of the manipulator is at the center of the track circle that is $x(0) = [1.0 \ 1.0]$ m, $\dot{x}(0) = [0.0 \ 0.0]$ m/s.

Because the tracking trajectory is the Cartesian coordinates in the workspace, the joint position of the 2-DOF manipulator should be converted to the Cartesian coordinates of the joint end in the workspace according to the Jacobian matrix $J(q)$:

$$J(q) = \begin{bmatrix} -l_1 \sin(q_1) - l_2 \sin(q_1 + q_2) & -l_2 \sin(q_1 + q_2) \\ l_1 \cos(q_1) + l_2 \cos(q_1 + q_2) & l_2 \cos(q_1 + q_2) \end{bmatrix}, \quad (35)$$

and taking the derivative of $J(q)$ yields

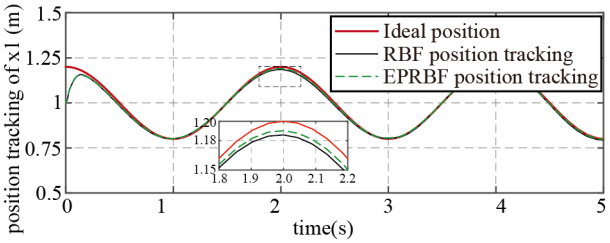
$$\dot{J}(q) = \begin{bmatrix} -l_1 \cos(q_1) - l_2 \cos(q_1 + q_2) & -l_2 \cos(q_1 + q_2) \\ -l_1 \sin(q_1) - l_2 \sin(q_1 + q_2) & -l_2 \sin(q_1 + q_2) \end{bmatrix} \dot{q}_1 + \begin{bmatrix} -l_2 \cos(q_1 + q_2) & -l_2 \cos(q_1 + q_2) \\ -l_2 \sin(q_1 + q_2) & -l_2 \sin(q_1 + q_2) \end{bmatrix} \dot{q}_2. \quad (36)$$

The number of hidden layers of each neural network is designed to be 7. Simultaneously, for the control scheme without the EPSO optimizer, the center of the Gaussian function c_j is selected as 7 points of the uniform distribution in [0, 12] and the width of the Gaussian function b_i = 5, and the initial weights of all the neural networks are selected as 0. The controller parameters are chosen as K = [30 0; 0 30], δ =0.2 and Λ = [15 0; 0 15]. In the adaptive law in Eq. (23), the parameters are selected as $\Gamma_k = \mathrm{diag}\{2.0\}$, $\Psi_k = \mathrm{diag}\{0.10\}$ and N_k = diag {5.0}.

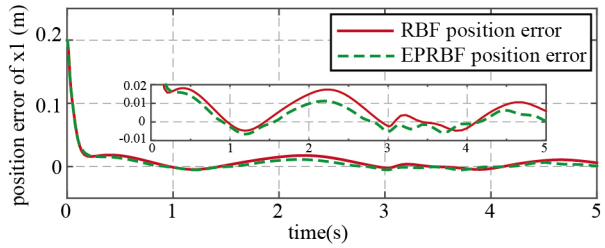
Regarding the EPSO optimizer, the initial values of the parameters of the EPSO optimizer are defined as follows: comprehensive coefficient C_{epso} = 2; space range from -10 to 25; number of particles N_{pop} = 63; number of iterations M = 40; and number of populations N = 10.

4.2 Simulation results

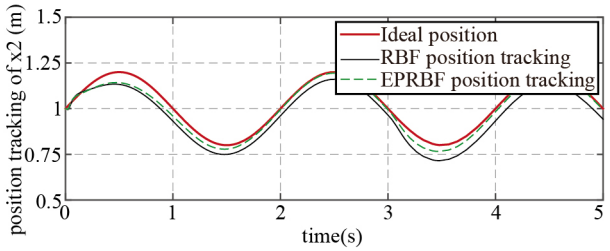
In this subsection, the comparison results between the control performance of the EPRBF controller and the RBF controller are analyzed. Among them, the RBF controller is the same neural network as in the above scheme without the EPSO



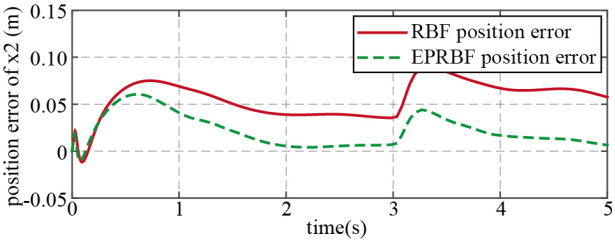
(a) Position tracking in the x1 direction



(b) Position errors in the x1 direction

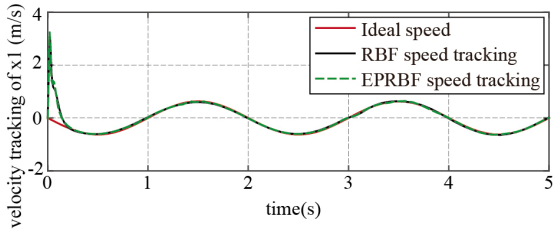


(c) Position tracking in the x2 direction

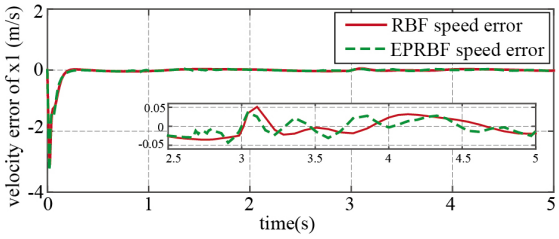


(d) Position errors in the x2 direction

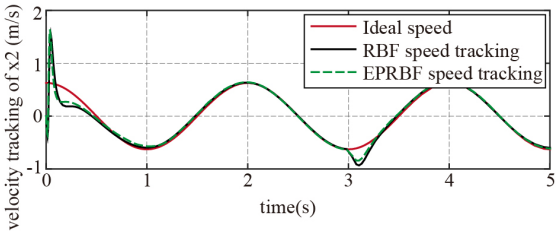
Fig. 5. Position trajectories of EPRBF and RBF by the fixed external load.



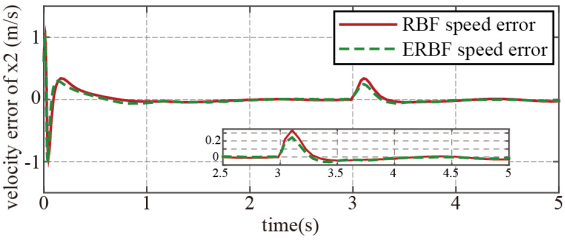
(a) Velocity tracking in the x1 direction



(b) Velocity errors in the x1 direction



(c) Velocity tracking in the x2 direction



(d) Velocity errors in the x2 direction

Fig. 6. Velocity trajectories of EPRBF and RBF by the fixed external load.

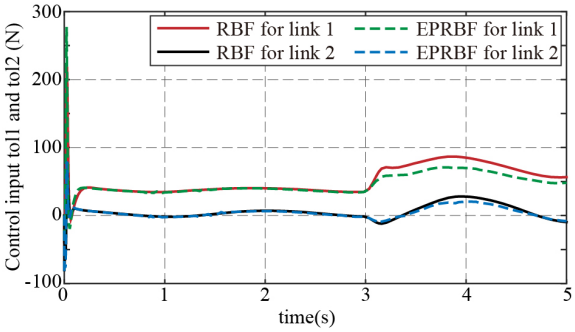


Fig. 7. Control input of link 1 and link 2 by the fixed external load.

optimizer. To test the performance of anti-interference, two external loads are considered: the fixed force p_l = 2 and p_l = sin ($2\pi t$) when time exceeds 3 s.

Simulation 1: For the first simulation, the position tracking results, speed tracking results, and joint output torque results of the two control schemes are shown in Figs. 5-8.

Fig. 5 shows the position tracking process and tracking errors of the two control methods on the x1 and x2-axes. Figs. 5(b) and (d) show that the position tracking error of the EPRBF controller is significantly less than that of the RBF controller after 0.4 s, and the disturbance error of the EPRBF controller is slightly smaller than that of the RBF controller when the manipulator is subjected to external interference. Fig. 6 shows the speed tracking process and speed errors of the two types of controllers in the x1 and x2-axes. Because the proportion of the velocity error in the compound error is small, the speed tracking performance of the controllers is similar. Fig. 7 shows the output torque curve of joint 1 and joint 2 as a function of time when the EPRBF and RBF control schemes are

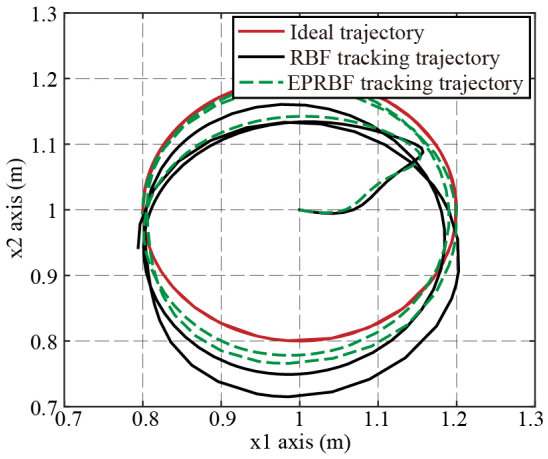
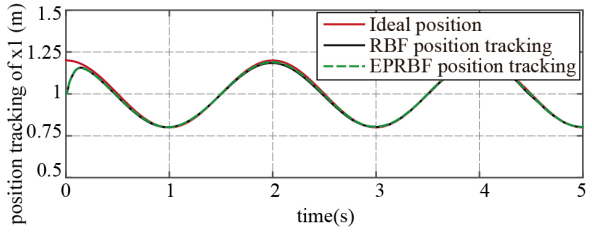


Fig. 8. Tracking the trajectory of the end of the manipulator.

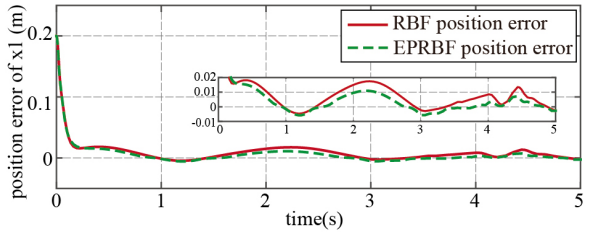
used, respectively. The curve shows that the output torque by the EPRBF controller has a faster convergence speed and smaller convergence errors with external disturbance than the RBF controller. Fig. 8 shows that the EPRBF controller made the end of the robotic manipulator reach the ideal trajectory faster, and the position tracking performance is significantly better than that of the RBF controller.

Simulation 2: In the second simulation, the time-varying force is considered, and the simulation results are shown in Figs. 9-11.

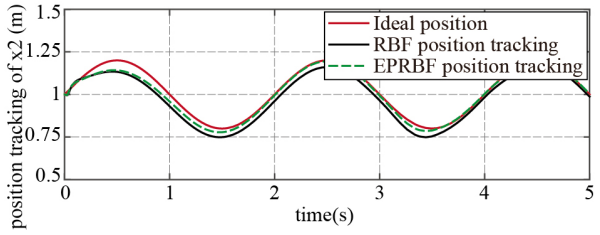
In Fig. 9, in the case in which a variable external disturbance is applied, the position error curves fluctuated twice after 3 s. The fluctuation of the position error curves and the convergence speed of the EPRBF control scheme are slightly better than those of the RBF control scheme. Fig. 10 shows that the fluctuations of the velocity in the EPRBF control scheme are



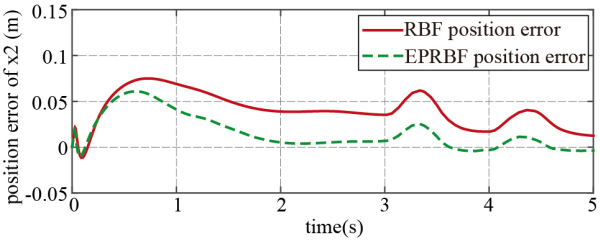
(a) Position tracking in the x1 direction



(b) Position errors in the x1 direction

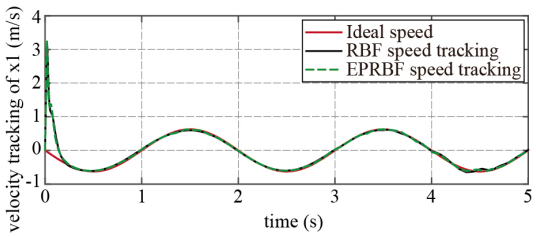


(c) Position tracking in the x2 direction

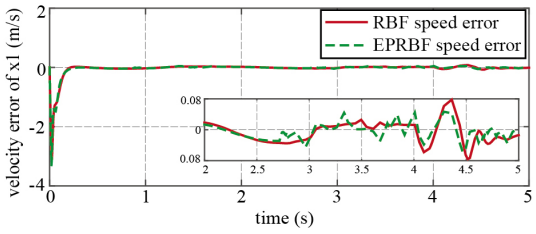


(d) Position errors in the x2 direction

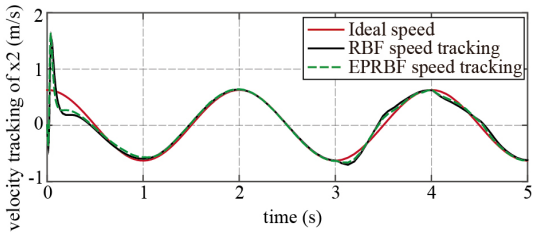
Fig. 9. Position trajectories of EPRBF and RBF by the time-varying external load.



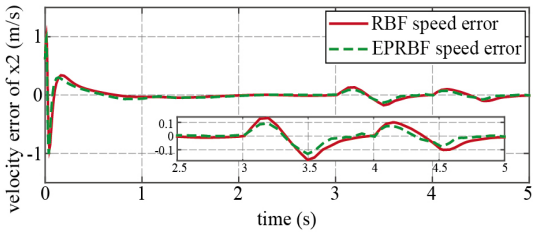
(a) Velocity tracking in the x1 direction



(b) Velocity errors in the x1 direction



(c) Velocity tracking in the x2 direction



(d) Velocity errors in the x2 direction

Fig. 10. Velocity trajectories of EPRBF and RBF by the time-varying external load.

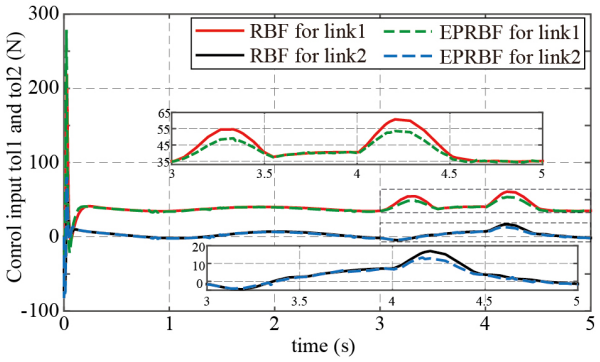


Fig. 11. Control input of link 1 and link 2 by the time-varying external load.

smaller than those of the RBF control scheme on the x1 and x2-axes. Fig. 11 shows that after the external disturbance is applied, the joint control torque has two significant fluctuations in the x1-axis direction and a significant fluctuation in the x2-axis direction. The joint control torque fluctuations in the EPRBF control scheme are all smaller than those of the RBF control scheme.

To summarize, the EPRBF controller proposed in this paper has better position tracking and speed tracking performance on the target trajectory, in addition to better anti-interference ability. Simultaneously, the EPRBF controller also has smaller control torque fluctuation with external interference.

4.3 Further analysis and discussion

In the previous section, the superiority of the proposed scheme over the RBF control scheme is verified through simulation results. To compare the performance of the RBF controller and EPRBF controller more specifically and accurately, the composite error $r(t)$ is used as the evaluation index for analysis. The position and velocity error curves obtained by the EPRBF and RBF controllers are equally distributed into 190 values by time and then substituted into Eq. (18). The composite error obtained by the RBF control scheme is 147.4373 and that of the EPRBF control scheme is 80.5362. The composite error of the EPRBF control scheme is 54.62 % of the RBF control scheme, and the optimal effect is remarkable. The hyperparameters b and c are shown in Table 1, respectively.

Using the EPRBF control scheme to improve the tracking accuracy of the manipulator required a longer optimization calculation time as the number of EPSO iterations increased. The composite error $r(t)$ is still used as the evaluation index. The position and velocity error curves obtained by EPRBF are equally distributed into 145 values by time, and the number of iterations is 20, 40, 50, 70, 80, 90, 100, 110, 130 and 150. The number of iterations, composite errors and calculation times are shown in Fig. 12.

In Fig. 12, when the number of iterations is less than 80, the optimized composite error is random as the number of iterations increased. When the number of iterations is more than 80, the composite error decreased as the number of iterations

Table 1. Hyperparameters of the RBF and EPRBF control schemes.

	RBF	EPRBF
c-M	[0 2 4 6 8 10 12; 0 2 4 6 8 10 12]	[-6.196 0 11.785 -0.131 -2.444 3.582 9.078; 0.035 -4.345 8.15 0.086 3.267 3.421 -5.383]
c-G	[0 2 4 6 8 10 12; 0 2 4 6 8 10 12]	[0.453 -1.872 3.324 2.932 0.561 0.562 2.084; 0.321 -1.359 2.807 7.568 0.659 0.301 2.12]
c-C	[0 2 4 6 8 10 12; 0 2 4 6 8 10 12; -6 -4 -2 0 2 4 6 ; -6 -4 -2 0 2 4 6]	[-4.41 -3.897 -4.846 -3.257 1.686 5.428 -1.906; 3.907 0.777 -9.515 10.295 -9.14 2.877 -2.272; 6.848 -5.096 1.747 2.785 -7.085 -3.533 10.562; 6.848 -5.096 1.747 2.785 -7.085 -3.533 10.562]
b	[5 5 5 5 5 5 5]	[3.891 14.989 8.082 21.546 3.259 18.170 11.24]

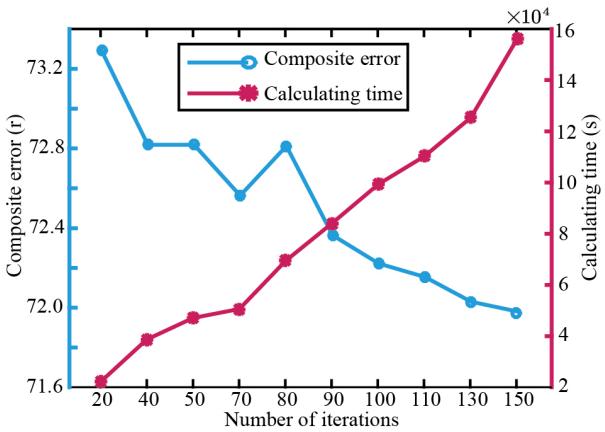


Fig. 12. Error and calculation time variations of the EPSO algorithm for different iteration times.

increased. Simultaneously, when the number of iterations increased from 100 to 150, the composite error decreased by 0.8842 %, but the calculation time increased by 57.2348 %. We can infer that better control performance and a relatively short calculation time can be obtained when the number of iterations is around 100.

In order to further fairly compare and illustrate the advantages of the proposed controller, three PSO optimization algorithms, including the modified Kalman particle swarm optimization (MKPSO) [39], the random weighted PSO (RandWPSO) [40], the black hole particle swarm optimization (BHPSO) [41], are used to optimize the RBF controller as optimizers for comparison. The value of compound error $r(t)$ mentioned in this paper is used as the evaluation index, and the calculation time of each control scheme is also compared. The aforementioned three PSO algorithms are replaced by the EPSO algorithm described in this paper and the number of iterations is set to 100. The 2-DOF manipulator model is used as the object for the simulation. The simulation results are shown in Fig. 13. Figs. 13(a) and (c) show that the EPSO optimizer enables the RBF controller to achieve the best optimization result, and the compound error obtained by the optimization is 72.2216. The MKPSO optimizer can no longer reduce the system compounding error after the number of iterations reach 18, and it fell into the local optimum the earliest. By contrast, the optimization effect of the RandWPSO is the worst. Fig. 13(b) shows

Table 2. The general specifications of the robotic manipulator.

Manufacturer	Elmo, Israel
DC power supply	48.0 V
Current	4.6 A
Rating	150 W
Range of operation for joint 2 and joint 3	±146°
Sampling time	100.0 μs
Switching frequency of the inverter	22.0 KHZ

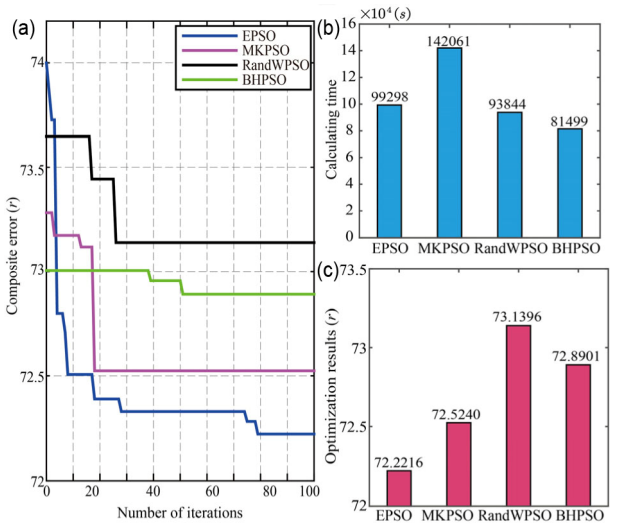


Fig. 13. Comparison of four different algorithms for uncertain robotic manipulator: (a) optimization processes of different optimization algorithms when the number of iterations is 100; (b) operation time of each optimization algorithm; (c) optimization results of various optimization algorithms.

that the sequencing of calculation time required for the four optimization algorithms is MKPSO > EPSO > RandWPSO > BHPSO, and there is not a substantial difference among the calculation times of the latter three algorithms. From the above comparison results, we can see that the control scheme proposed in this paper can obtain better performance in trajectory tracking control of uncertain manipulators.

5. Experimental studies

To demonstrate the performance of the proposed control scheme, experiments are conducted to test the feasibility of a practical application based on a 6-DOF manipulator. Figs. 14 and 15 show the experimental test setup configuration and the control block diagram. The platform consisted of four parts: a monitoring computer, DC regulated power supply, controller and 6-DOF manipulator. In the joint module of the manipulator, a 20000-line incremental encoder is used for the angle detection of the joint torque motor of the manipulator, and a 17-bit absolute encoder is used to detect the joint angle. The general specifications of the robotic manipulator are listed in Table 2 and the specifications of the controller are listed in Table 3. Compared with the traditional complex controller, this paper

Table 3. The general specifications of the controller.

Processor	TMS320F28335 DSP
Signal input	IO: 3 channels
Signal output	D/A converter: 4 channels 16 bit
Encoder	Digital incremental encoder interface
Maximum input frequency	20 MHz
Communication interface	TTL 485 CAN

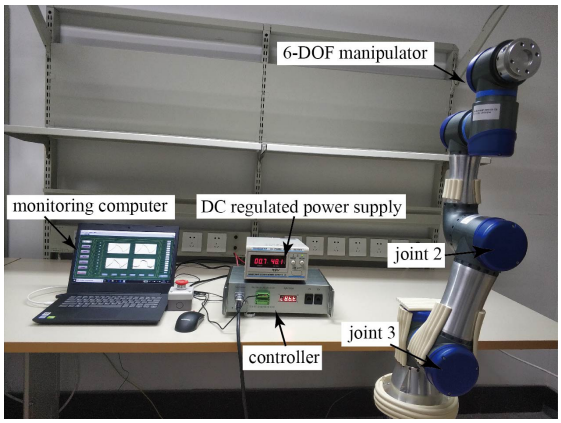


Fig. 14. Experimental test setup.

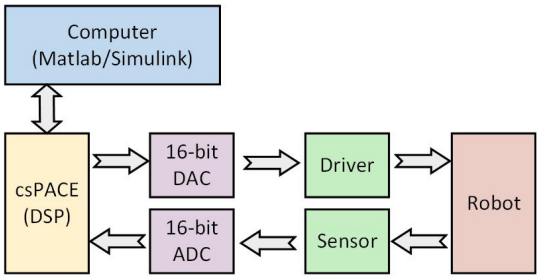


Fig. 15. The control block diagram of robotic manipulator system.

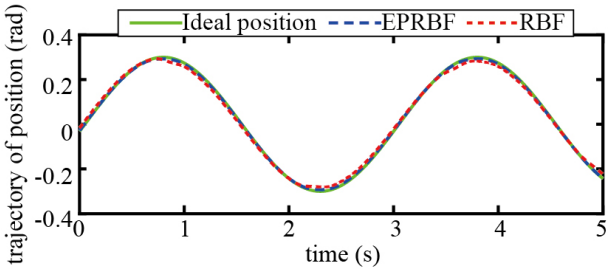
uses a rapid controller prototyping cSPACE, which can provide an efficient real-time control algorithm for the experimental platform [42]. Using the C2000 DSP embedded target toolbox of MATLAB R2018a, the code can be automatically generated online, and the rapid prototyping and embedded system development can be realized. According to the requirements of code composer studio (CCS) project files, the Simulink program automatically generates the C code needed by TI DSP. On this basis, download the C code to run on the DSP target board and complete the real-time implementation of the algorithm.

The following specific steps for the implementation of the experiment are given:

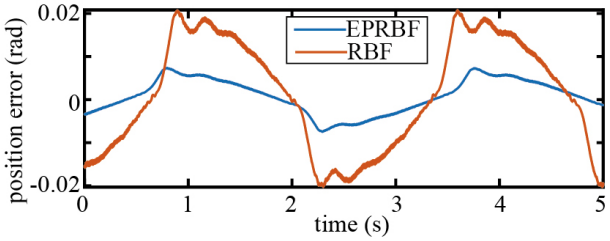
Step 1: The proposed control algorithm is offline programmed using MATLAB/Simulink (MATLAB R2018a or a higher version).

Step 2: Through the cSPACE control platform, the MATLAB/Simulink program directly converted into C codes.

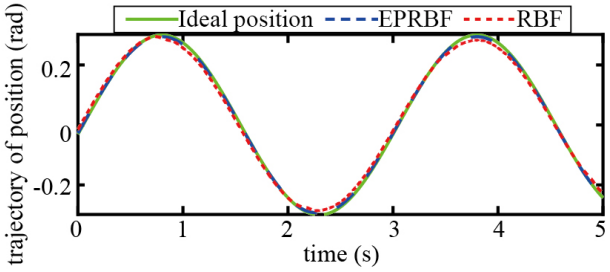
Step 3: In the CCS environment, the converted C codes are



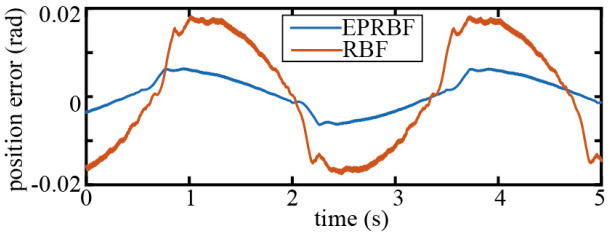
(a) Position tracking of joint 2



(b) Position errors of joint 2



(c) Position tracking of joint 3



(d) Position errors of joint 3

Fig. 16. Experiment results of the EPRBF and RBF control scheme.

downloaded to the DSP control board of the controller through the simulator.

Step 4: The proposed control algorithm is employed by a DSP processor in the form of C codes with a sampling time of 5 ms.

The experimental results are shown in Fig. 16. In Figs. 16(a) and (c), compared with the ideal joint position curve, the performance of EPRBF controller is significantly better than that of RBF controller. Fig. 16(b) shows that the maximum trace error of joint 2 remained under $\pm 75\times10^{-4}$ rad by EPRBF controller and under $\pm 207\times10^{-4}$ rad by RBF controller. The relative error of joint 2 reduced from 6.9 % to 2.5 %. Fig. 16(d) shows that the relative error of joint 3 reduced from 6.00 % to 2.02 %. The experiment experimental results verify that the EPRBF controller has the accuracy position tracking ability.

6. Conclusion

In this paper, an EPRBF control scheme for improving the tracking accuracy and anti-interference ability of the robotic manipulator is proposed. The hyperparameters in the RBFNN controller based on local approximation are continuously optimized by the EPSO optimizer. To verify the superiority of the proposed control scheme, a two-DOF manipulator loaded with different external disturbances is simulated. In the case of the fixed force, the composite error decreased from 147.4373 to 80.5362, and the error decreased to the original 54.62 %. Furthermore, the control performance for different iterations is observed and the simulation results show that when the number of iterations is set to 100, better control performance and less calculation time can be obtained. To verify the feasibility of the proposed control scheme, experimental studies are conducted and the results show that the maximum relative error remained under 2.5 % in the manipulator control system. The simulation and experimental results demonstrate the superiority and effectiveness of the proposed EPSO optimizer, which improves the tracking accuracy and anti-interference performance of robotic manipulators. In future works, the number of layers of NNs and the number of neurons in the controller will be mainly studied to coordinate control performance and computation time.

Acknowledgments

This work is partially supported by the National Natural Science Foundation of China (11972343, 91848202), the Funded and Supported by National Key R&D Program of China (No.2016YFE0205000).

Nomenclature

T	: Target index
N_{pop}	: Population size
LB	: Lower bounds of the search space
UB	: Upper bounds of the search space
C_{epos}	: Combined index
$Rand$	: A random number between [0 1]
X_p	: Sample space
c	: Center vector
b	: Width of the Gaussian function
N	: Number of nodes in the hidden layer
w	: Weight of the output layer
J	: Number of nodes in the output layer
y	: Output of the neural network
Φ_N	: Number of hidden layer nodes
w^*	: Ideal bounded weights
$\varepsilon(x)$	: Reconstruction error
$\hat{w}$	: Estimated value of the ideal weight
$\underline{q}$	: Joint angular position vectors
$\dot{q}$	: Joint angular velocity vectors
$\ddot{q}$	: Joint angular acceleration vectors

$M(q)$: Symmetric and positive definite inertia matrix
$C(q, \dot{q})$: Centrifugal and Coriolis forces matrix
$G(q)$: Gravity vector
τ : Torque input vector
τ_d : Unknown disturbances
$M_0(q)$: Nominal value of $M(q)$
$C_0(q, \dot{q})$: Nominal value of $C_0(q, \dot{q})$
$G_0(q)$: Nominal value of $G(q)$
E_M : Model error of $M(q)$
E_C : Model error of $C(q, \dot{q})$
E_G : Model error of $G(q)$
x : The position and orientation of the end effector
$J(q)$: Jacobian matrices
$M_{SNN}(q)$, $C_{DNN}(q, \dot{q})$, $G_{SNN}(q)$: Output items of the RBFNN
$X_d(t)$: Ideal trajectory
$\dot{x}_d(t)$: Ideal velocities
$\ddot{x}_d(t)$: Ideal accelerations
$e(t)$: Position tracking error
$\dot{x}_r(t)$: Neural network modeling speed
$r(t)$: Composite error
Λ : Ratio constant
δ : Defined positive constant
c-M, c-G, c-C : Center point vectors of the hidden layer neurons
P : Parameter vector of the manipulator
p_l : External disturbance
l_1, l_2 : Length of link 1 and link 2
K, δ : Controller parameters

References

[1] Z. Zhang et al., High precision control and deep learning-based corn stand counting algorithms for agricultural robot, *Autonomous Robots*, 44 (7) (2020) 1289-1302.

[2] H. Y. Sai et al., Adaptive nonsingular fixed-time sliding mode control for uncertain robotic manipulators under actuator saturation, *ISA Transactions*, 21 (2021) 00266-4.

[3] X. Y. Yao, H. F. Ding and M. F. Ge, Synchronization control for multiple heterogeneous robotic systems with parameter uncertainties and communication delays, *Journal of the Franklin Institute*, 356 (16) (2019) 9713-9729.

[4] K. Zheng, Y. Hu and B Wu, Trajectory planning of multi-degree-of-freedom robot with coupling effect, *Journal of Mechanical Science and Technology*, 33 (1) (2019) 413-421.

[5] F. Meng, L. Zhao and J. Yu, Backstepping based adaptive finite-time tracking control of manipulator systems with uncertain parameters and unknown backlash, *Journal of the Franklin Institute*, 357 (16) (2020) 11281-11297.

[6] S. N. Kumpati and P. Kannan, Identification and control of dynamical systems using neural networks, *IEEE Transactions on Neural Networks*, 1 (1) (1990) 4-27.

[7] J. J. E. Slotine and W. Li, On the adaptive control of robot manipulators, *The International Journal of Robotics Research*, 6 (3) (1987) 49-59.

[8] K. S. Narendra and S. Mukhopadhyay, Adaptive control of nonlinear multivariable systems using neural networks, *Proceedings of 32nd IEEE Conference on Decision and Control*, IEEE (1993) 3066-3071.

[9] L. Jin et al., Robot manipulator control using neural networks: a survey, *Neurocomputing*, 285 (2018) 23-34.

[10] Y. Guo and J. Liu, Neural network based adaptive dynamic surface control for flight path angle, *2012 51st IEEE Conference on Decision and Control (CDC)*, IEEE (2012) 5374-5379.

[11] X. Bei, Z. Tianping and Y. Yuequan, Adaptive neural network control of flexible robotic with unmodeled dynamics and time-varying output constraints, *2017 36th Chinese Control Conference (CCC)*, IEEE (2017) 3331-3336.

[12] W. He and Y. Dong, Adaptive fuzzy neural network control for a constrained robot using impedance learning, *IEEE Transactions on Neural Networks and Learning Systems*, 29 (4) (2017) 1174-1186.

[13] C. Lee and D. An, Reinforcement learning and neural network-based artificial intelligence control algorithm for self-balancing quadruped robot, *Journal of Mechanical Science and Technology*, 35 (2021) 1-16.

[14] Z. Jiang et al., Retina-based pipe-like object tracking implemented through spiking neural network on a snake robot, *Frontiers in Neurorobotics*, 13 (2019) 29.

[15] S. I. Han and J. M. Lee, Fuzzy echo state neural networks and funnel dynamic surface control for prescribed performance of a nonlinear dynamic system, *IEEE Transactions on Industrial Electronics*, 61 (2) (2013) 1099-1112.

[16] H. R. Nohooji, I. Howard and L. Cui, Neural network adaptive control design for robot manipulators under velocity constraints, *Journal of the Franklin Institute*, 355 (2) (2018) 693-713.

[17] A. Azizi, Applications of artificial intelligence techniques to enhance sustainability of industry 4.0: design of an artificial neural network model as dynamic behavior optimizer of robotic arms, *Complexity* (2020) 1-10.

[18] A. Azizi, A case study on computer-based analysis of the stochastic stability of mechanical structures driven by white and colored noise: utilizing artificial intelligence techniques to design an effective active suspension system, *Complexity* (2020) 1-8.

[19] H. Ismkhan, Ik-means−+: an iterative clustering algorithm based on an enhanced version of the k-means, *Pattern Recognition*, 79 (2018) 402-413.

[20] S. S. Yu et al., Two improved k-means algorithms, *Applied Soft Computing*, 68 (2018) 747-755.

[21] J. Dong et al., Orthogonal least squares based center selection for fault-tolerant RBF networks, *Neurocomputing*, 339 (2019) 217-231.

[22] J. Moody and C. J. Darken, Fast learning in networks of locally-tuned processing units, *Neural Computation*, 1 (2) (1989) 281-294.

[23] Z. Liu et al., An analytical approach to fast parameter selection of gaussian RBF kernel for support vector machine, *J. Inf. Sci. Eng.*, 31 (2) (2015) 691-710.

[24] Y. Hu et al., An eigenvector based center selection for fast training scheme of RBFNN, *Information Sciences*, 428 (2018) 62-75.

[25] S. Khan et al., A fractional gradient descent-based RBF neural network, *Circuits, Systems, and Signal Processing*, 37 (12) (2018) 5311-5332.
[26] J. Ye and M. Lu, Design optimization of domes against instability considering joint stiffness, *Journal of Constructional Steel Research*, 169 (2020) 105757.
[27] L. Amamra et al., The concepts of genetics applied to a reinforced concrete cantilever beam optimization, *Sustainability and Automation in Smart Constructions*, Springer Cham, New York (2020) 309-315.
[28] C. Chen et al., Fuzzy adaptive control particle swarm optimization based on TS fuzzy model of maglev vehicle suspension system, *Journal of Mechanical Science and Technology*, 34 (1) (2020) 43-54.
[29] Z. H. Zhan et al., Adaptive particle swarm optimization, *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39 (6) (2009) 1362-1381.
[30] F. Wang et al., A hybrid particle swarm optimization algorithm using adaptive learning strategy, *Information Sciences*, 436 (2018) 162-177.
[31] T. T. Ngo, A. Sadollah and J. H. Kim, A cooperative particle swarm optimizer with stochastic movements for computationally expensive numerical optimization problems, *Journal of Computational Science*, 13 (2016) 68-82.
[32] E. J. Hartman, J. D. Keeler and J. M. Kowalski, Layered neural networks with Gaussian hidden units as universal approximations, *Neural Computation*, 2 (2) (1990) 210-215.
[33] J. Park and I. W. Sandberg, Universal approximation using radial-basis-function networks, *Neural Computation*, 3 (2) (1991) 246-257.
[34] L. Yu et al., Trajectory switching control of robotic manipulators based on RBF neural networks, *Circuits, Systems, and Signal Processing*, 33 (4) (2014) 1119-1133.
[35] W. Zeng and C. Wang, Learning from NN output feedback control of robot manipulators, *Neurocomputing*, 125 (2014) 172-182.
[36] H. Yang and J. Liu, An adaptive RBF neural network control method for a class of nonlinear systems, *IEEE/CAA Journal of Automatica Sinica*, 5 (2) (2018) 457-462.
[37] L. Sciavicco and B. Siciliano, Modelling and control of robot manipulators, *Measurement Science and Technology*, 11 (12) (2000) 1828-1829.
[38] S. G. Shuzhi, C. C. Hang and L. C. Woon, Adaptive neural network control of robot manipulators in task space, *IEEE Transactions on Industrial Electronics*, 44 (6) (1997) 746-752.
[39] H. Lei, B. Chen, Y. Liu and Y. Lv, Modified Kalman particle swarm optimization: application for trim problem of very flexible aircraft, *Engineering Applications of Artificial Intelligence*, 100 (2021) 104176.
[40] G. Yuelin and D. Yuhong, A new particle swarm optimization algorithm with random inertia weight and evolution strategy, *2007 International Conference on Computational Intelligence and Security Workshops (CISW 2007)* (2007) 199-203.
[41] A. Harb, H. Kassem and K. Ghorayeb, Black hole particle swarm optimization for well placement optimization, *Computational Geosciences*, 24 (6) (2020) 1979-2000.
[42] K. H. Hong et al., Rapid prototyping of DSP algorithms on VLIW TMS320C6701 DSP, *Microprocessors and Microsystems*, 26 (7) (2002) 311-324.

Appendix

Substituting Eqs. (15)-(17) into Eq. (13) yields

$$\begin{aligned}&\{[\{\Theta\}^T\cdot\{Q(q)\}]+E_M(q)\}\ddot{x}\\&+\{[\{A\}^T\cdot\{Z(z)\}]+E_c(z)\}\dot{x}\\&+[\{B\}^T\cdot\{H(q)\}]+E_G(q)=F_x.\end{aligned}\tag{A.1}$$

Substituting the control rule in Eq. (19) into Eq. (A.1) yields

$$\begin{aligned}&\{[\{\Theta\}^T\cdot\{Q(q)\}]+E_M(q)\}\ddot{x}\\&+\{[\{A\}^T\cdot\{Z(z)\}]+E_c(z)\}\dot{x}\\&+[\{B\}^T\cdot\{H(q)\}]+E_G(q)\\&=\left[\left\{\hat{\Theta}\right\}^T\cdot\{Q(q)\}\right]\ddot{x}_r+\left[\left\{\hat{A}\right\}^T\cdot\{Z(z)\}\right]\dot{x}_r\\&+\left[\left\{\hat{B}\right\}^T\cdot\{H(q)\}\right]+Kr+k_a sat(r).\end{aligned}\tag{A.2}$$

Substituting $\dot{x}=\dot{x}_r=r$ and $\ddot{x}=\ddot{x}_r=\dot{r}$ into Eq. (A.2) yields

$$\begin{aligned}&\{[\{\Theta\}^T\cdot\{Q(q)\}]+E_M(q)\}(\ddot{x}_r-\dot{r})\\&+\{[\{A\}^T\cdot\{Z(z)\}]+E_c(z)\}(\dot{x}_r-r)\\&+[\{B\}^T\cdot\{H(q)\}]+E_G(q)\\&=\left[\left\{\hat{\Theta}\right\}^T\cdot\{Q(q)\}\right]\ddot{x}_r+\left[\left\{\hat{A}\right\}^T\cdot\{Z(z)\}\right]\dot{x}_r\\&+\left[\left\{\hat{B}\right\}^T\cdot\{H(q)\}\right]+Kr+k_a sat(r).\end{aligned}\tag{A.3}$$

Simplifying Eq. (A.3) yields

$$\begin{aligned}&\{[\{\Theta\}^T\cdot\{Q(q)\}]+E_M(q)\}\dot{r}+Kr\\&+\{[\{A\}^T\cdot\{Z(z)\}]+E_c(z)\}r+k_a sat(r)\\&=\left[\left\{\tilde{\Theta}\right\}^T\cdot\{Q(q)\}\right]\ddot{x}_r+\left[\left\{\tilde{A}\right\}^T\cdot\{Z(z)\}\right]\dot{x}_r\\&+\left[\left\{\tilde{B}\right\}^T\cdot\{H(q)\}\right]+E.\end{aligned}\tag{A.4}$$

Then, substituting Eqs. (15) and (16) into Eq. (A.4), the tracking error equation is given by

$$\begin{aligned}&M_x(q)\dot{r}+C_x(q,\dot{q})r+Kr+k_a sat(r)\\&=\left[\left\{\tilde{\Theta}\right\}^T\cdot\{Q(q)\}\right]\ddot{x}_r+\left[\left\{\tilde{A}\right\}^T\cdot\{Z(z)\}\right]\dot{x}_r\\&+\left[\left\{\tilde{B}\right\}^T\cdot\{H(q)\}\right]+E.\end{aligned}\tag{A.5}$$

Huayang Sai received a B.E. degree in School of Mechanical and Electronic Engineering form Northwest A&F University, Yangling, China, in 2018. He is now a Ph.D. candidate in Mechanical Engineering and Automation, Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Science, China. He is also currently pursuing the degree with the College of Optoelectronics, University of Chinese Academy of Sciences, Beijing. His current research interests include medical robot, robot impedance control.

Zhenbang Xu received the B.E. degree from the Department of Theoretical and Applied Mechanics, Chinese Academy of Sciences University, Hefei, China, in 2005, and the Ph.D. degree from the Chinese Academy of Sciences University, in 2010, where he is currently with the Changchun Institute of Optics, Fine Mechanics and Physics. His research interests include space intelligent robot, multi-dimensional precision adjustment mechanism, space structure dynamics, and microvibration control.

Ce Xu received a B.E. degree in Qingdao University of Science and Technology, China, in 2016. He is now a Ph.D. candidate in University of Chinese Academy of Sciences, China and Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences. His current research interests include mechanics analysis, kinematics, and dynamics of space robots.

Kai Wang received the B.E. degree from the Department of Mechanical Engineering, North China Electric Power University, Baoding, China, in 2018. He is currently pursuing the M.E. degree in Mechanical Engineering and Automation with the Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences, China. He is also currently pursuing the degree with the College of Optoelectronics, University of Chinese Academy of Sciences, Beijing. His current research interests include research on kinematics, dynamics, and control of continuum robots.

Xiaoming Wang, born in Ji'an, Jilin Province, China, holds a Doctor's degree. She obtained a Bachelor's degree from Jilin University in 2014 and a Doctor's degree from Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences in 2019. Her research field is dynamic modeling and simulation analysis of space robots.

Lin Zhu received the B.E. degree from Northeast Normal University, Changchun, China, in 2018. She is currently pursuing the Ph.D. degree in Adavanced Optical Imaging Technology and Space Camera Optical Design Research in Optical Engineering with the Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences, China.