Reference modification for trajectory tracking using hybrid offline and online neural networks learning

Jiangang Li¹ · Youhua Huang¹ · Ganggang Zhong¹ · Yanan Li²

Received: 29 November 2021 / Accepted: 5 February 2022 / Published online: 7 March 2022
© The Author(s) 2022

Abstract

In this paper, we propose a hybrid offline/online neural networks learning method, which combines complementary advantages of two types of neural networks (NNs): deep NN (DNN) and single-layer radial basis function NN (RBFNN). Firstly, after analyzing the mechatronic system's model, we select reasonable features as the input of the DNN to learn the inverse dynamic characteristics of the closed-loop system offline, so as to establish the mapping between the desired trajectory and the reference trajectory of the system. The trained DNN is used to generate a new reference trajectory and compensate for the tracking error in advance, which can speed up the convergence of online learning control based on RBFNN. This reference trajectory is further modified iteratively when the tracking task is repeated. For this purpose, a single-layer RBFNN model is established, and an online learning algorithm is developed to update the RBFNN parameters. The proposed hybrid offline/online NN method can improve the tracking performance of mechatronic systems by modifying the reference trajectory on top of the baseline controller without affecting the system stability. To verify the effectiveness of this method, we conduct experiments on a piezoelectric drive platform.

Keywords Deep neural network · Trajectory tracking · RBF neural network · Mechatronic systems

1 Introduction

Trajectory tracking is a fundamental problem in control of mechatronic systems, e.g., robot manipulators and piezoelectric actuators (PEAs). In most cases, there are uncertainties and external disturbances, such as friction, sensor noise and variations of payload in the operations of these systems with nonlinear dynamics. A sequence of control methods were proposed to solve these issues, such as

adaptive control [1], sliding mode control [2], learning control [3, 4], and neural network control [5, 6]. Moreover, mechatronic systems with imperfections is an universal control problem. In [7], a control strategy to ensure the optimal working conditions was proposed, which focused on the effects of using chaotic vibrational signals to excite the hidden dynamics of the imperfect system. In [8], the authors focused on a paradigmatic example of imperfect electromechanical structure and developed a control method to ensure coil rotation based on the excitation of the hidden dynamics induced by imperfections, characterizing its influence on the characteristics of the control signal and the power provided to the structure. Imperfections also play an important role in the realization of robust chaos generators based on simple circuits. In [9], a strategy for estimating hidden dynamics parameters was designed and synchronization of imperfect chaotic circuits was achieved. Compared to active research on advanced control approaches in academia, classical linear controllers such as PID control are still playing a crucial part in industries for the sake of implementation simplicity. However, it is well known that PID controller behaves poorly on complex

✉ Yanan Li
yl557@sussex.ac.uk

Jiangang Li
jiangang_lee@hit.edu.cn

Youhua Huang
19s153189@stu.hit.edu.cn

Ganggang Zhong
ganggang_zhong@hit.edu.cn

¹ School of Mechanical Engineering and Automation, Harbin Institute of Technology, Shenzhen 518055, China

² Department of Engineering and Design, University of Sussex, Brighton BN1 9RH, UK

trajectories and on systems with nonlinear dynamics [10]. Therefore, it is interesting to develop a control approach that is built on top of the off-the-shelf linear controllers but improves the tracking performance. Such a control approach has two significantly favorable features. First, most of the controllers provided by the manufacturers do not allow the users to modify low-level position controller but provides access to tunable parameters and a reference trajectory. In a position control task, the desired trajectory is the trajectory predefined for the mechatronic system to actually track. The reference trajectory can be obtained by using the trajectory generator module to modify the desired trajectory and is used as the input signal of the closed-loop mechatronic system. Second, without modifying the available control architecture, the system stability can be in general ensured. In this regard, many state-of-the-art controllers that design the control input such as [11–19] are not applicable.

To cope with disturbances and imperfections of mechatronic systems, a lot of research effort has been made on modifying the reference trajectory to improve the tracking performance on top of an available feedback control system. A large group of these works is iterative learning control (ILC) that improves the performance of trajectory tracking with repetition of a same task and using knowledge from previous iterations [20–22]. Although learning convergence can be proved in rigor, the information about the system learned by ILC cannot be transferred to another task, similar to adaptive control [23].

As mechatronic systems generally have complicated dynamics, which are influenced by uncertainties [24], there is ample motivation to investigate the effectiveness of machine learning in the control of mechatronic systems [25]. Some researchers were attracted by the excellent capabilities of deep neural networks (DNNs) in function approximation and thus revisited the idea of constructing an NN model for mechatronic systems [26], especially the inverse compensation control based on NN model [27–29]. In [30], a polynomial fitting model based on NN was proposed to describe the inverse dynamics of hysteresis in PEA. As a feedforward compensation module, the model is combined with a single neurogenic adaptive proportional integral differential controller to reduce the trajectory tracking error caused by hysteresis in piezoelectric drive mechatronic system. Different from the traditional control framework based on NN inverse model to approximate the open-loop dynamics and modify the control signal of the plant, the offline learning control framework proposed in this paper uses DNN to approximate the inverse dynamic characteristics of the closed-loop mechatronic system and uses the trained DNN as a trajectory generator to modify the reference trajectory, so that the tracking error can be compensated for in advance without changing the structure

and stability of the baseline controller. Although the offline learning method can approximate the inverse dynamics of the closed-loop mechatronic system, the DNN model is still subject to modeling error, and the tracking control accuracy needs to be further improved based on online learning.

The online learning control framework based on iterative learning in this paper is suitable for repetitive tasks and can suppress unknown uncertainties. Compared to DNN, single-hidden-layer radial basis function neural networks (RBFNNs) have the advantages of simple structure and high computational efficiency. RBFNN is simple to implement in real time, and its learning convergence and the resultant closed-loop system stability can be strictly analyzed [31, 32]. The control schemes based on RBFNN in the closed-loop control system mainly include supervisory control, model reference adaptive control, self-tuning control, etc. In [33], for a class of nonlinear systems with unknown parameters and bounded disturbances, RBFNN combined with single-parameter direct adaptive control was designed to overcome the problems caused by unknown dynamics and external disturbances in nonlinear systems. The traditional control methods based on RBFNN modify the control signal of the controlled plant, and the parameters of the RBFNN need to be updated all the time [34, 35]. Different from these works, the online learning control framework based on RBFNN proposed in this paper uses iterative method to update the parameters of the RBFNN and modify the reference trajectory until the tracking error is reduced below a target threshold. The advantages of the method proposed in this paper are as follows: (1) For the repetitive trajectory, the repetitive interference and error in the system can be suppressed. (2) It does not change the structure of the baseline controller and will not affect the stability of the closed-loop system. Thus, it can be easily applied to commercial control systems.

Based on the above discussions, this paper will investigate reference trajectory modification for mechatronic systems, by integrating a DNN for offline learning and a single-layer RBFNN for online learning. First of all, DNN is offline-trained to approximate the inverse dynamics model of mechatronic systems, and the trained DNN is used to obtain the modified reference trajectory as the input of the closed-loop mechatronic system or further modified by online learning of RBFNN. Then, we propose the online learning control framework-based RBFNN, and combined with Lyapunov function, we design the learning law of RBFNN and prove the stability of the system. The offline NN learning method learns the inverse dynamics of the closed-loop system and speeds up the online learning, which can compensate for tracking error in advance. The online learning NN method can deal with uncertainties and

disturbances and thus achieve precise trajectory tracking control.

The main contribution of this paper is the hybrid offline/online learning control framework, which combines complementary advantages of DNN and a single-layer RBFNN. On the one hand, we propose the offline learning control framework with DNN as a reference trajectory generator, which is transferrable and can be used to conduct a new tracking task. Offline learning can provide an initial reference trajectory for online learning and speed up the convergence of RBFNN parameters; on the other hand, we propose online learning control framework with RBFNN to iteratively modify the reference trajectory generated by DNN as the input signal of the closed-loop mechatronic system, and prove its convergence.

The remaining structure of this paper is as below: Section 2 shows the system dynamics and transforms the control problem into mathematical models and introduces the proposed tracking control method based on hybrid offline/online NN. Sections 3 and 4 elaborate the processes of online and offline learning, respectively. Section 5 presents the results of the experiments. Section 6 concludes this work.

2 System description and control strategy

2.1 System description

According to [36–38], the schematic model of piezoelectric actuator illustrates a reversible transformation from electrical charge to mechanical energy, as shown in Fig. 1, where H , C , T_{em} and x denote the hysteresis effect, the capacitance, the electromechanical transducer and the output displacement of piezoelectric actuator, respectively.

The dynamic equation of piezoelectric actuator can be expressed as:

$$m_z \ddot{x} + b_z \dot{x} + k_z x = T_{em}(u_{in} - u_h) \quad (1)$$

Fig. 1 Schematic model of piezoelectric actuator

where u_{in} denotes the input voltage, u_h the voltage due to the hysteresis, and m_z , b_z , and k_z are the mass, damping, and stiffness of the ceramic, respectively.

In practice, there are external disturbances exerting on the piezoelectric drive system besides the nonlinear hysteresis. In order to contain these effects, the piezoelectric drive system can be described as

$$m \ddot{x} + b \dot{x} + kx + v_n + v_d = u_{in} \quad (2)$$

where v_n and v_d represent all the nonlinear effects and external disturbances, $m = m_z/T_{em}$, $b = b_z/T_{em}$, and $k = k_z/T_{em}$.

From Eq. (2), the dynamic model of a mechatronic system (piezoelectric actuator) can be generalized to a second-order system,

$$M(x) \ddot{x} + C(x, \dot{x}) \dot{x} + G(x) + D(x, \dot{x}) = u \quad (3)$$

where u denotes the control input and x denotes the position. Design the control input u as a linear state feedback controller, which is commonly used in a motion controller provided by the manufacturer, i.e.,

$$u = -K[(\dot{x} - \dot{x}_d) + \alpha(x - x_r)] \quad (4)$$

where $\alpha > 0$, and K , x_d , and x_r denote control gain, desired trajectory, and reference trajectory, respectively. When $x_r = x_d$, u is a conventional PD controller that can be rewritten as

$$u = -K(\dot{e} + \alpha e), e = x - x_d \quad (5)$$

where e denotes the trajectory tracking error. Suppose that x_d is a constant and ignore the disturbance vector $D(x, \dot{x})$ and gravity vector $G(x)$; then, we can obtain the formula as below by considering Eqs. (5) and (3):

$$M(x) \ddot{x} + [C(x, \dot{x}) + K] \dot{x} + K\alpha e = 0 \quad (6)$$

By checking the above equation, it is straightforward to confirm the system stability with $x \rightarrow x_d$ when $t \rightarrow \infty$. Nevertheless, the PD controller cannot achieve $x \rightarrow x_d$ if the disturbance $D(x, \dot{x})$ and gravity $G(x)$ have dramatic effects on the dynamics of the system or if x_d is a trajectory. Therefore, we will design x_r which considers the uncertainties of the dynamics. In particular, the closed-loop dynamics can be written as below with Eq. (4).

$$M(x) \ddot{x} + [C(x, \dot{x}) + K] \dot{x} + K\alpha x + G(x) + D(x, \dot{x}) = K\dot{x}_d + K\alpha x_r \quad (7)$$

The design of reference trajectory x_r includes two learning processes: offline learning (generating x_{r_off}) and online learning (generating x_{r_on}), for which the details are discussed in the following.

2.2 Control strategy

The hybrid offline and online learning trajectory tracking control framework proposed in this paper is shown in Fig. 2.

Offline learning refers to learning the inverse dynamic model with a DNN so it modifies the desired trajectory to a new reference trajectory x_{r_off} . The online learning part will take the output of the offline learning as the initial reference trajectory. Then, it will use single-layer RBFNNs to obtain reference trajectory x_{r_on} to locally compensate for the unmodeled dynamics and uncertainties to further improve the tracking performance.

In this control framework, the control system can be divided into three parts. The first part is a closed-loop mechatronic system, which contains an inaccessible controller. Its input is the reference trajectory x_r modified by DNN and RBFNNs, and the output is the actual trajectory x . The controller in the closed-loop mechatronic system ensures that the system can achieve a certain precision of trajectory tracking and has good feedback properties, that is, good robustness, repeatability and anti-interference properties. The second part is the DNN trajectory generation module based on offline learning. Its input is the desired trajectory x_d , and the output is the modified reference trajectory x_{r_off} . The DNN with powerful approximation capability can be used to learn the inverse dynamics of the closed-loop mechatronic system, making the desired trajectory and the actual trajectory of the whole system approach the identity mapping. The trained DNN is used as an additional trajectory generation module to modify the reference trajectory to reduce the tracking error of the system. The third part is the RBFNN trajectory modification module based on online learning. Its input is the reference trajectory x_{r_off} and the output is the reference trajectory x_{r_on} . The RBFNN trajectory modification module is for repetitive trajectories, iteratively modifying the reference trajectory x_{r_on} to approximate an ideal reference trajectory $x_{r_on}^*$. Thus, the repetitive disturbance is compensated for in advance. These two learning processes are elaborated in detail in the following two sections, respectively.

3 Offline learning

3.1 Control strategy

The offline NN learning part in the hybrid offline/online control framework is shown in Fig. 3.

First, the DNN trajectory generation module needs to be trained offline using training data. The offline learning control framework is divided into training phase and testing phase, and the information of actual trajectory x is used as the training input of DNN and the desired trajectory x_d is the training output.

The transfer function of a closed-loop mechatronic system can be defined as

$$G(s) = \frac{X(s)}{X_r(s)} \quad (8)$$

where $X_r(s)$ and $X(s)$ denote x_r and x in s -domain, respectively. If the reference trajectory generator has a transfer function $G^{-1}(s) = \frac{X_r(s)}{X(s)}$, then we will obtain $X(s) = X_d(s)$, i.e., perfect trajectory tracking. By selecting reasonable features as input, we can train a DNN model to approximate $G^{-1}(s)$. This DNN can be then used to generate a reference trajectory x_{r_off} with the input of a new desired trajectory x_d .

3.2 Feature selection

More states are required by the DNN for training to better approximate the characteristics of the system. Nevertheless, the increase of states leads to a large dimension of input, which requires superabundant training data as a result of the curse of dimension [39]. Therefore, we should choose the related features reasonably to minimize the dimension of the input.

In the framework of offline learning, training DNN is to learn the inverse dynamics characteristics of the closed-loop mechatronic system and establish the mapping relationship between the desired trajectory and the reference trajectory, i.e., approximating $G^{-1}(s) = \frac{X_r(s)}{X(s)}$, to achieve zero tracking error, i.e., $x_d = x$. It is noticed that x_r is

Fig. 2 Control framework of the proposed hybrid offline and online neural networks learning

Fig. 3 Control framework of the proposed offline neural networks learning

related to $(x, \dot{x}, \ddot{x})$ and $\dot{x}_d$, i.e., $\dot{x}$ by analyzing dynamics of the closed-loop system mentioned in Eq. (7). Hence, we should select the triple $(x, \dot{x}, \ddot{x})$ as the DNN training input. Due to the delay of the mechatronic system, the current x_r will affect the future $(x, \dot{x}, \ddot{x})$, so the future information $(x, \dot{x}, \ddot{x})$ could also be added to training input to improve the performance [40].

3.3 Training and testing data

From the structure of Eq. (7), we know there are mass/inertial, Coriolis and centrifugal terms, gravity and disturbance in the dynamic model of the system, which could also include significant coupling among different axes. Hence, one of the key factors to determine the effectiveness of DNN training is whether the information can fully represent the properties of a mechatronic system. In this sense, random nonuniform rational B-spline surface (NURBS) curve can be used.

We generate the random NURBS trajectory by using random control points [41], which are composed of independent variable vector and dependent variable vector, and the independent variable vector $\mathbf{t}$ is as follows

$$\mathbf{t} = [t_0, t_0 + \Delta t_1, \dots, t_0 + \sum_{i=1}^n \Delta t_i] \quad (9)$$

We set $\Delta t_i = 0.01 + \text{rand}(0.02, 0.04)$, where $\text{rand}(0.02, 0.04)$ represents a random number between $0.02 \sim 0.04$, and we set the dependent variable observation vector $\mathbf{x}$ to satisfy the normal distribution with mean and standard deviation of 25 and 10, respectively.

According to the independent variable vector $\mathbf{t}$ and the dependent variable vector $\mathbf{x}$, the random control points sequence $y(\mathbf{x}, \mathbf{t})$ will be obtained as the training or testing trajectories. In the experiments, according to the movement range of the closed-loop piezoelectric drive system, the movement trajectory is mapped to $0 \sim 60 \mu\text{m}$, and zero-phase filtering is performed to remove the peak point of the

speed exceeding the limit, and finally, the training and testing trajectory is obtained.

3.4 Training of DNN

The above offline learning belongs to supervised learning, which builds a mapping between the input and output of the system with the knowledge of the desired output and given input [42]. The NN is usually trained with the backpropagation (BP) algorithm whose effect is to minimize the error between the desired and actual outputs of the NN.

Denote W_{ij}^l as the weight which connects the j -th neuron of the $l-1$ -th layer and the i -th neuron of the l -th layer and b_i^l as the bias of the i -th neuron in the l -th layer. Then, the input of the i -th neuron from the l -th layer can be described as below

$$\text{net}_i^l = \sum_{j=1}^{s_{l-1}} W_{ij}^l h_j^{l-1} + b_i^l, h_i^l = f(\text{net}_i^l) \quad (10)$$

where s_l denotes the neuron number in the l -th layer and f represents an activation function.

We define an error function as below

$$E = \frac{1}{m} \sum_{i=1}^m E(i), E(i) = \frac{1}{2} \sum_{k=1}^n (y_k(i) - y_k^*(i))^2 \quad (11)$$

where m and n denote the number of the training data groups and the outputs, respectively; y_k , y_k^* denote actual and desired outputs. After that, we use the gradient descent method to update the weights as below

$$W_{ij}^l = W_{ij}^l - \beta \frac{\partial E}{\partial W_{ij}^l}, b_i^l = b_i^l - \beta \frac{\partial E}{\partial b_i^l} \quad (12)$$

where $\beta > 0$.

4 Online learning

4.1 Control strategy

While offline learning in the previous section establishes an inverse dynamics model that can generate a reference trajectory to improve the tracking performance, unmodeled dynamics or uncertainties may exist so the reference trajectory should be further modified online. For this purpose, in this section we derive an online learning algorithm using single-layer RBFNNs.

The online learning RBFNN control framework proposed in this paper is shown in Fig. 4. In theory, RBFNN can fit a continuous function with arbitrary accuracy. When only one RBFNN is used, it is bound to learn the noise and disturbance mean value of the whole trajectory and cannot further reduce the error of each point. In this paper, for repetitive trajectories, a large number of small RBFNNs are used to fit the local dynamics model of each trajectory point. The control framework uses the idea of iteration to update the weight of RBFNNs according to the error between the desired value $x_d(k)$ and the actual value $x(k)$ obtained from the last run at each trajectory point and uses updated RBFNNs to generate the reference trajectory point for the next run, so that the generated reference trajectory x_{r_on} constantly approaches the ideal reference trajectory $x_{r_on}^*$.

4.2 RBF neural networks

A RBFNN has an input layer, a hidden layer and an output layer [43]. In the input layer, the input signals $z = [z_1, z_2, \dots, z_n]$ are moved directly to the next layer. The hidden layer consists of an array of computing units, which are referred to as hidden nodes. Each neuron in the hidden layer is activated by a radial basis function. The output of the hidden layer is computed as follows:

$$s_j(z) = \exp\left(-\frac{(\|z - c_j\|)^2}{2b_j^2}\right), j = 1, \dots, m \quad (13)$$

where m is the number of hidden nodes, $c_j = [c_{j1}, \dots, c_{jn}]$ is the center vector, b_j denotes the standard deviation of the j -

th radial basis function, and s_j is the Gaussian function. In the output layer, the output signal is a linearly weighted combination as follows:

$$y(z) = \sum_{j=1}^m w_j s_j(z) \quad (14)$$

where w_j is the weight for the j -th node.

In [44], it is shown that for any continuous function $f(z) : \Omega_z \rightarrow R$, where $\Omega_z \subset R^q$ is a compact set, when the node number m is sufficiently large, there exists an ideal constant weight W , such that for each $\epsilon^* > 0$:

$$f(z) = \sum_{j=1}^m w_j s_j(z) = W^T S(z) + \epsilon(z), \forall z \in \Omega_z \quad (15)$$

where $|\epsilon(z)| < \epsilon^*$ is the approximation error.

4.3 Design of learning law

In this subsection, we explain how to develop an online learning algorithm to update the weights of the RBFNNs. Let us consider a desired controller with the knowledge of the system dynamics:

$$u^* = -K(\dot{e} + \alpha e) + M\ddot{x}_d + C\dot{x}_d + G + D \quad (16)$$

where $\ddot{x}_e = \ddot{x}_d - \alpha\dot{e}$ and $\dot{x}_e = \dot{x}_d - \alpha e$. Note that the arguments of M , C , G , and D are omitted, where no confusion is caused. By defining the sliding error

$$\varepsilon = \dot{e} + \alpha e \quad (17)$$

and substituting Eq. (16) into Eq. (3), we have desired closed-loop dynamics

$$M\dot{\varepsilon} + (C + K)\varepsilon = 0 \quad (18)$$

It is easy to see from the above equation that $\varepsilon \rightarrow 0$ and thus $e \rightarrow 0$ when $t \rightarrow \infty$, indicating that trajectory tracking is achieved. Therefore, we design the controller (4) the same as in Eq. (16), i.e., $u = u^*$, which leads to

$$x_{r_on}^* = \frac{1}{\alpha K} (M\ddot{x}_d + C\dot{x}_d + G + D) + x_d \quad (19)$$

The above equation indicates the ideal reference trajectory $x_{r_on}^*$ can achieve trajectory tracking without error under

Fig. 4 Control framework of the proposed online neural networks learning

the PD controller. However, we note that the dynamics parameters M , C , G and disturbance D are unknown. This motivates us to use a single-hidden-layer NN to approximate $x_{r_on}^*$, i.e.,

$$x_{r_on}^* = W^T S(Z) + \epsilon \quad (20)$$

where W denotes unknown ideal weight, S denotes an activation function, Z is NN input and ϵ is the approximation error. Therefore, the reference trajectory can be written as

$$x_{r_on} = \hat{W}^T S(Z) \quad (21)$$

where $\hat{W}$ is the actual weight that needs to be updated. Based on Lyapunov theory that will be elaborated in the following section, we design an update law of $\hat{W}$ as below:

$$\Delta \hat{W} = -\alpha K Q \varepsilon^T S(Z) \quad (22)$$

where Q is a positive-definite matrix. $\Delta(\cdot) = (\cdot)(t) - (\cdot)(t-T)$ where T is the time duration of a task and $(\cdot) = 0$ when its argument is smaller than 0.

4.4 Online learning convergence

In this subsection, we show that the proposed learning algorithm guarantees convergence. Substituting Eq. (21) into Eq. (7) and defining $\tilde{W} = \hat{W} - W$, we obtain

$$M\dot{\varepsilon} + (C + K)\varepsilon = K\alpha(\tilde{W}^T S - \epsilon) \quad (23)$$

Let us choose a Lyapunov function candidate

$$J = J_\varepsilon + J_W = \frac{1}{2} \varepsilon^T M \varepsilon + \frac{1}{2} \int_{t-T}^t \text{vec}^T(\tilde{W}) Q^{-1} \text{vec}(\tilde{W}) d\tau \quad (24)$$

where $\text{vec}(\cdot)$ is the vectorization operation. Considering the first term in J , we have

$$\dot{J}_\varepsilon = \varepsilon^T M \dot{\varepsilon} + \frac{1}{2} \varepsilon^T \dot{M} \varepsilon \quad (25)$$

Considering the skew-symmetric property, i.e.,

$$z^T \dot{M} z = 2z^T C z, \forall z \quad (26)$$

and Eq. (23), we have

$$\dot{J}_\varepsilon = \varepsilon^T (M\dot{\varepsilon} + C\varepsilon) = \varepsilon^T [-K\varepsilon + K\alpha(\tilde{W}^T S - \epsilon)] \quad (27)$$

By taking the integral of the above equation from $t-T$ to t , we have

$$\Delta J_\varepsilon = \int_{t-T}^t \varepsilon^T [-K\varepsilon + K\alpha(\tilde{W}^T S - \epsilon)] d\tau \quad (28)$$

Now, we consider the second term in J and have

$$\begin{aligned} \Delta J_W &= J_W(t) - J_W(t-T) \\ &= \int_{t-T}^t (\text{vec}^T(\tilde{W})(t) Q^{-1} \text{vec}(\tilde{W})(t) \\ &\quad - \text{vec}^T(\tilde{W})(t) Q^{-1} \text{vec}(\tilde{W})(t-T) \\ &\quad + \text{vec}^T \tilde{W}(t) Q^{-1} \text{vec}(\tilde{W})(t-T) \\ &\quad - \text{vec}^T(\tilde{W})(t-T) Q^{-1} \text{vec}(\tilde{W})(t-T)) d\tau \\ &= \frac{1}{2} \int_{t-T}^t (2 \text{vec}^T(\tilde{W})(t) - \text{vec}^T(\Delta \tilde{W})) Q^{-1} \text{vec}(\Delta \tilde{W}) d\tau \\ &\leq \int_{t-T}^t \text{vec}^T(\tilde{W})(t) Q^{-1} \text{vec}(\Delta \tilde{W}) d\tau \end{aligned} \quad (29)$$

where $\Delta \tilde{W} = \tilde{W}(t) - \tilde{W}(t-T)$. Substituting update law in Eq. (22) into the above inequality, we obtain

$$\Delta J_W \leq \alpha \int_{t-T}^t \varepsilon^T K \tilde{W}^T(t) S d\tau \quad (30)$$

Combining Eqs. (28) and (30), we have

$$\Delta J = \Delta J_\varepsilon + \Delta J_W \leq \int_{t-T}^t -\varepsilon^T K(\varepsilon - \alpha\epsilon) d\tau \quad (31)$$

By Ineqs. (31), we have

$$J(t) - J(t-nT) \leq \int_{t-nT}^t -\varepsilon^T K(\varepsilon - \alpha\epsilon) d\tau \quad (32)$$

where n is the number of iterations. By setting $t = nT$, we have

$$J(nT) - J(0) \leq \int_0^{nT} -\varepsilon^T K(\varepsilon - \alpha\epsilon) d\tau \quad (33)$$

which leads to

$$\int_0^{nT} \varepsilon^T K(\varepsilon - \alpha\epsilon) d\tau \leq J(0) - J(nT) \leq J(0) \quad (34)$$

By the definition of J in Eq. (24), we know that $J(0)$ is bounded, so the left-hand side of the above inequality is also bounded. When $n \rightarrow \infty$, we have $\varepsilon^T K(\varepsilon - \alpha\epsilon) \rightarrow 0$. As ϵ can be made arbitrarily small with a large number of RBFNN nodes, ε becomes arbitrarily small which indicates almost perfect trajectory tracking.

4.5 Parameter initialization of RBFNN

Parameters w_j , c_j , and b_j in Eq. (15) should be in an effective field of mapping, otherwise, the RBFNN will not work properly. However, it is toilsome and impractical to choose the best parameters manually. To solve this problem, gradient decent method is used to initialize the parameters.

Since the reference trajectory should be close to the desired trajectory, we use RBFNN to fit the desired trajectory so as to initialize the parameters for online learning. In particular, the desired trajectory is approximated by the RBFNN as below:

$$x_d = W^T S(x) + \epsilon \quad (35)$$

The predicted output is presented as:

$$\hat{x}_d = \hat{W}^T S(x) \quad (36)$$

The error function is defined as

$$E(t) = \frac{1}{2} (x_d - \hat{x}_d)^2 \quad (37)$$

Then, using the gradient decent method, we have

$$\begin{aligned} \Delta w_j(t) &= -\gamma \frac{\partial E}{\partial w_j} = \gamma (x_d - \hat{x}_d) s_j(x) \\ \Delta b_j(t) &= -\gamma \frac{\partial E}{\partial b_j} = \gamma (x_d - \hat{x}_d) w_j s_j(x) \frac{\|x - c_j\|^2}{b_j^3} \\ \Delta c_{ji}(t) &= -\gamma \frac{\partial E}{\partial c_j} = \gamma (x_d - \hat{x}_d) w_j s_j(x) \frac{x_j - c_j}{b_j^2} \end{aligned} \quad (38)$$

5 Experiments

5.1 Experimental platform

To validate the effectiveness of the proposed hybrid offline and online learning trajectory tracking control method, a piezoelectric drive platform is used in experiments, as shown in Fig. 5. The piezoelectric drive platform is mainly composed of four parts: piezoelectric controller, PEA, real-time simulation controller, and a computer. The PEA used

Fig. 5 Closed-loop PEA experiment platform

in the platform is a cylindrical low-voltage PEA PSt150/10/60VS15 of the Harbin Core Tomorrow Science and Technology Co., Ltd., and its physical parameters are given in Table 1. The piezoelectric controller is the E53.B servo controller of the Harbin Core Tomorrow Science and Technology Co., Ltd., equipped with an SGS displacement sensor with sensitivity of 6um/V and measurement accuracy of 0.05um. The real-time simulation controller is the DS1103 PPC Controller Board of dSPACE GmbH.

In order to evaluate the performance of trajectory tracking control method proposed in this paper, tracking error e is defined, and the control objective is to reduce e . For tracking error e , the average relative error, maximum absolute error and root mean square error are defined as follows:

$$e = x - x_d \quad (39)$$

$$e_{\max} = \max(\|x - x_d\|) \quad (40)$$

$$e_{rms} = \sqrt{\frac{1}{n} \sum_{k=1}^n \|x(k) - x_d(k)\|^2} \quad (41)$$

$$e_{raver} = \frac{1}{n} \sum_{k=1}^n \frac{\|x(k) - x_d(k)\|}{x_d(k)} \times 100\% \quad (42)$$

In the experiment, the training and testing trajectories are generated by Eq. (9), and part of the trajectory is shown in Fig. 6. The proposed control framework and the corresponding experimental implementation are shown in Fig. 7.

Table 1 Physical parameters of PEA

Parameters	Values
Step response time	15 ms
Resolution	2 nm
Nominal stroke	60 um
Nominal thrust	2300 N
Stiffness	35 N/um

Fig. 6 NURBS trajectory

Fig. 7 The experiments block diagram of the proposed control framework

5.2 Offline learning experiment

According to [40], since nonlinear autoregressive neural network (NARX) has good-fitting ability for time series and allows efficient calculation, we use NARX as DNN trajectory generator. To test the offline learning control framework in Fig. 3, the experiment is performed as follows. i) The desired trajectory is input into the closed-loop PEA, and the corresponding actual trajectory is recorded. ii) According to the analysis of the feature selection and the experimental test, the actual state $(x, \dot{x}, \ddot{x})$ at the current time and the next 8 times are selected as the input of the neural network, and the desired trajectory x_d at the current time is used as the output to train the DNN to approach $G^{-1}(s) = \frac{X_r(s)}{X(s)}$. iii) The trained DNN module is connected in series in front of the reference trajectory storage of the closed-loop PEA, and the desired trajectory is input to the DNN module to generate a reference trajectory.

The structure of NARX is shown in Fig. 8, and the training parameters of the neural networks are shown in Table 2.

Fig. 8 Neural networks structure

Table 2 Training parameters of the neural networks

Parameters	Values
TrainFcn	'trainscg'
Minimum gradient	1e-10
Epochs	5000
Train goal	1e-7
Learning rate	0.05
Momentum factor	0.9
Input delays	0
Feedback delays	5

Under the offline learning framework as shown in Fig. 3, the desired trajectory, reference trajectory generated by DNN and actual trajectory of the closed-loop PEA are shown in Fig. 9. The DNN trajectory generator modifies the desired trajectory to obtain the reference trajectory, so that the actual trajectory of the system can track the desired trajectory.

Based on the trained DNN, the following three trajectory tracking control methods are compared: PID, linear active disturbance rejection control (LADRC) [45], feedforward compensation control based on DNN inverse model (Forward-DNN) [27], DNN trajectory generation control method based on offline learning (Offline-DNN). The parameters of PID controller are $Kp = 0.3$, $Ki = 1$, $Kd = 0.05$, and the parameters of LADRC are $b_0 = 0.3$, $w_0 = 25$, $w_c = 70$. The feedforward inverse compensation control uses DNN to establish the inverse model of the open-loop PEA as the feedforward compensator, combined with PID feedback controller. Here, we use the NURBS trajectory generated by Eq. (9) as the testing desired trajectory. The tracking errors between the desired trajectory and the actual trajectory under the three control frameworks are shown in Fig. 10 and Table 3.

Fig. 9 Trajectory of closed-loop PEA under offline learning. **(a)** Whole trajectory. **(b)** Local zoom trajectory

As shown in Table 3, after adding the offline-trained DNN trajectory generator to the closed-loop PEA, the maximum tracking error is reduced from $1.4668\mu\text{m}$ to $0.3718\mu\text{m}$, which is a reduction of 74.6%. The root mean square error is reduced from $0.4867\mu\text{m}$ to $0.1255\mu\text{m}$, a reduction of 74.2%. It can also be seen from Table 3 that the Offline-DNN has a lower trajectory tracking error than PID, LADRC and Forward-DNN.

5.3 Online learning experiment

To test the effectiveness of the online learning method based on RBFNN as shown in Fig. 4, we use the same desired trajectory in the offline learning experiment, and the experiment is performed as follows. (i) Input the desired trajectory x_d as the initial reference trajectory and initialize the weight parameters of RBFNNs. (ii) Run the closed-loop system with the reference trajectory. (iii) Calculate the error between the desired trajectory and the actual trajectory; if the error is less than the target threshold, stop updating the weight of RBFNNs, and save

Fig. 10 Tracking error under different trajectory tracking control methods

Table 3 Tracking error under different control methods

	PID	LADRC	Forward-DNN	Offline-DNN
$e_{\max}$ (um)	1.4668	0.8181	0.5055	0.3718
e_{rms} (um)	0.4867	0.2913	0.1924	0.1255
e_{raver} (%)	2.10	1.33	0.96	0.73

the current reference trajectory to the reference trajectory storage as the final input signal of the closed-loop PEA; otherwise, the weight of RBFNNs is updated according to the error. (iv) Generate the next reference trajectory with the updated RBFNN module. (v) Repeat step (ii). The parameters in Eq. (22) are set as: $\alpha = 110$, $K = 0.04$, $Q = 0.0001$.

The experimental results of online learning-based RBFNN are shown in Fig. 11. As shown in Fig. 11b, the root mean square error in each iteration reduces as the iteration number increases, and the value is $0.0067\mu\text{m}$ in the 10th iteration. After the tracking error reaches the target threshold, the weights of RBFNNs no longer need to be updated online, and the subsequent reference trajectory x_{r_on} will not change.

5.4 Hybrid learning experiment

In the hybrid offline/online learning control method as shown in Fig. 2, we use the reference trajectory x_{r_off} generated by offline-trained DNN as the initial reference trajectory in online learning control, which is referred to as hybrid learning. We compare the tracking error convergence of the following three trajectory tracking control methods: iterative learning control (ILC) [46], which directly alters the control signal to the plant, online learning and hybrid learning. The ILC uses a P-type learning

Fig. 11 Tracking error of closed-loop PEA under online learning. (a) Tracking error. (b) RMSE tracking error

law, and the learning rate is 0.8. The results of tracking error convergence are shown in Fig. 12 and Table 4.

As shown in Fig. 12 and Table 4, in the hybrid learning control method, the offline learning speeds up the learning process with a smaller tracking error in the first iteration. After several iterations, the tracking errors in both online learning and hybrid learning conditions converge, with a slight difference. Moreover, the trajectory tracking accuracy of offline learning is further improved by online learning. After 10 iterations, the maximum tracking error is reduced from $0.3718\mu\text{m}$ to $0.0171\mu\text{m}$, which is a reduction of 95.4%, and the root mean square error is reduced from $0.1255\mu\text{m}$ to $0.0050\mu\text{m}$, a reduction of 96.0%. It can also be seen from the results that the hybrid learning has faster convergence and lower tracking error than ILC.

6 Conclusion

In this paper, we proposed two types of reference trajectory modification methods based on neural networks for a mechatronic system, which were used to learn inverse

Table 4 Tracking error under different control methods in the 10th iteration

	ILC	Online learning	Hybrid learning
$e_{\text{max}}(\mu\text{m})$	0.5631	0.0260	0.0171
$e_{\text{rms}}(\mu\text{m})$	0.2242	0.0067	0.0050
$e_{\text{raver}}(\%)$	1.20	0.03	0.02

dynamics and thus to achieve precise trajectory tracking. They were combined to formulate a hybrid offline and online learning paradigm, which includes the following features: (i) it is applicable to various mechatronic systems (a piezoelectric drive platform in this paper); (ii) it adopts an embedded feedback controller whose access is not required; (iii) the offline-trained DNN model owns the generalization capability which means it can be applied to different tasks with new trajectories; (iv) the use of small RBFNNs makes online learning efficient and robust to system uncertainties; and finally, (v) the offline learning can benefit online learning by providing a “good” initial

Fig. 12 Comparison of tracking error convergence under different control methods after 10 iterations. (a) MAX tracking error. (b) RMSE tracking error. (c) Raver tracking error

reference trajectory. Note that offline learning and online learning can be used separately for different cases, e.g., online learning can be used when a task is repetitive and offline learning can be used when a system is not affected significantly by unknown disturbances. Our future works include testing the proposed approach on a robot manipulator with high nonlinearity.

Acknowledgements This work was supported by Shenzhen Science and Technology Program under Grant GXWD20201230155427003-20200821171505003 and in part by National Natural Science Foundation of China under Grant U1913213.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Isermann R (1996) Modeling and design methodology for mechatronic systems. *IEEE/ASME Trans Mechatron* 1(1):16–28
- Yang J, Li S, Yu X (2012) Sliding-mode control for systems with mismatched uncertainties via a disturbance observer. *IEEE Trans Industr Electron* 60(1):160–169
- Arimoto S, Kawamura S, Miyazaki F (1984) Bettering operation of dynamic systems by learning: a new control theory for servomechanism or mechatronics systems. In: The 23rd IEEE conference on decision and control, pp. 1064–1069. IEEE
- Li J, Wang Y, Li Y, Luo W (2020) Reference trajectory modification based on spatial iterative learning for contour control of two-axis NC systems. *IEEE/ASME Trans Mechatron* 25(3):1266–1275
- He W, Huang B, Dong Y, Li Z, Su C-Y (2017) Adaptive neural network control for robotic manipulators with unknown dead-zone. *IEEE Trans Cybern* 48(9):2670–2682
- Zhang P, Wu Z, Dong H, Tan M, Yu J (2020) Reaction-wheel-based roll stabilization for a robotic fish using neural network sliding mode control. *IEEE/ASME Trans Mechatron* 25(4):1904–1911
- Bucolo M, Buscarino A, Famoso C, Fortuna L, Frasca M (2019) Control of imperfect dynamical systems. *Nonlinear Dyn* 98(4):2989–2999
- Bucolo M, Buscarino A, Famoso C, Fortuna L, Frasca M (1984) Smart control of imperfect electromechanical systems. In: 2019 IEEE international conference on systems, man and cybernetics (SMC), pp. 1882–1886 (2019). IEEE
- Bucolo M, Buscarino A, Famoso C, Fortuna L, Gagliano S (2021) Imperfections in integrated devices allow the emergence of unexpected strange attractors in electronic circuits. *IEEE Access* 9:29573–29583
- Li Q, Qian J, Zhu Z, Bao X, Helwa MK, Schoellig AP (2017) Deep neural networks for improved, impromptu trajectory tracking of quadrotors. In: 2017 IEEE International Conference on Robotics and Automation (ICRA), pp. 5183–5189. IEEE
- Yang C, Li Z, Cui R, Xu B (2014) Neural network-based motion control of an underactuated wheeled inverted pendulum model. *IEEE Trans Neural Netw Learn Syst* 25(11):2004–2016
- Tran M-D, Kang H-J (2015) A local neural networks approximation control of uncertain robot manipulators. In: International conference on intelligent computing, pp. 551–557. Springer
- He W, Chen Y, Yin Z (2015) Adaptive neural network control of an uncertain robot with full-state constraints. *IEEE Trans Cybern* 46(3):620–629
- Van Cuong P, Nan WY (2016) Adaptive trajectory tracking neural network control with robust compensator for robot manipulators. *Neural Comput Appl* 27(2):525–536
- He W, Dong Y, Sun C (2016) Adaptive neural impedance control of a robotic manipulator with input saturation. *IEEE Trans Syst Man Cybern Syst* 46(3):334–344. <https://doi.org/10.1109/TSMC.2015.2429555>
- Li T, Duan S, Liu J, Wang L, Huang T (2016) A spintronic memristor-based neural network with radial basis function for robotic manipulator control implementation. *IEEE Trans Syst Man Cybern Syst* 46(4):582–588. <https://doi.org/10.1109/TSMC.2015.2453138>
- Wang F, Chao Z-q, Huang L-b, Li H-y, Zhang C-q (2017) Trajectory tracking control of robot manipulator based on RBF neural network and fuzzy sliding mode. *Cluster Computing*, 1–11
- Jian Y, Huang D, Liu J, Min D (2019) High-precision tracking of piezoelectric actuator using iterative learning control and direct inverse compensation of hysteresis. *IEEE Trans Industr Electron* 66(1):368–377
- Li S, Shao Z, Guan Y (2019) A dynamic neural network approach for efficient control of manipulators. *IEEE Trans Syst Man Cybern Syst* 49(5):932–941. <https://doi.org/10.1109/TSMC.2017.2690460>
- Tayebi A (2004) Adaptive iterative learning control for robot manipulators. *Automatica* 40(7):1195–1203
- Schoellig AP, Mueller FL, D'Andrea R (2012) Optimization-based iterative learning for precise quadcopter trajectory tracking. *Auton Robot* 33(1–2):103–127
- Bristow DA, Tharayil M, Alleyne AG (2006) A survey of iterative learning control. *IEEE Control Syst Mag* 26(3):96–114
- Abdelatti M, Yuan C, Zeng W, Wang C (2018) Cooperative deterministic learning control for a group of homogeneous nonlinear uncertain robot manipulators. *Sci China Inf Sci* 61(11):1–19
- Cully A, Mouret J-B (2016) Evolving a behavioral repertoire for a walking robot. *Evol Comput* 24(1):59–88
- Martinez-Cantin R (2017) Bayesian optimization with adaptive kernels for robot control. In: 2017 IEEE international conference on robotics and automation (ICRA), pp. 3350–3356. IEEE
- Hinton GE, Osindero S, Teh Y-W (2006) A fast learning algorithm for deep belief nets. *Neural Comput* 18(7):1527–1554
- Napole C, Barambones O, Calvo I, Velasco J (2020) Feedforward compensation analysis of piezoelectric actuators using artificial neural networks with conventional PID controller and single-neuron PID based on hebb learning rules. *Energies* 13(15):3929
- Zhang X, Tan Y, Su M, Xie Y (2010) Neural networks based identification and compensation of rate-dependent hysteresis in piezoelectric actuators. *Physica B* 405(12):2687–2693

29. Napole C, Barambones O, Derbeli M, Silaa MY, Calvo I, Velasco J (2020) Tracking control for piezoelectric actuators with advanced feed-forward compensation combined with PI control. In: Multidisciplinary Digital Publishing Institute Proceedings, vol. 64, p. 29
30. Liang Y, Xu S, Hong K, Wang G, Zeng T (2019) Neural network modeling and single-neuron proportional-integral-derivative control for hysteresis in piezoelectric actuators. *Measurement and Control -London- Institute of Measurement and Control-* 52(9–10):002029401986684
31. Zeng W, Wang Q, Liu F, Wang Y (2016) Learning from adaptive neural network output feedback control of a unicycle-type mobile robot. *ISA Trans* 61:337–347
32. Kong L, Li D, Zou J, He W (2020) Neural networks-based learning control for a piezoelectric nanopositioning system. *IEEE/ASME Trans Mechatron*, 1–1
33. Yang H, Liu J (2018) An adaptive RBF neural network control method for a class of nonlinear systems. *IEEE/CAA J Autom Sinica* 5(2):457–462
34. Liang H, Liu G, Zhang H, Huang T (2020) Neural-network-based event-triggered adaptive control of nonaffine nonlinear multi-agent systems with dynamic uncertainties. *IEEE Trans Neural Netw Learn Syst* 32(5):2239–2250
35. Slama S, Errachdi A, Benrejeb M (2018) Model reference adaptive control for mimo nonlinear systems using RBF neural networks. In: 2018 international conference on advanced systems and electric technologies (IC_ASET), pp. 346–351 . IEEE
36. Goldfarb M, Celanovic N (1997) Modeling piezoelectric stack actuators for control of micromanipulation. *IEEE Control Syst Mag* 17(3):69–79
37. Adriaens H, De Koning WL, Banning R (2000) Modeling piezoelectric actuators. *IEEE/ASME Trans Mechatron* 5(4):331–341
38. Liaw HC, Shirinzadeh B, Smith J (2007) Enhanced sliding mode motion tracking control of piezoelectric actuators. *Sens Actuators A* 138(1):194–202
39. Cruz F, Simas Filho E, Albuquerque M, Silva I, Farias C, Gouvêa L (2017) Efficient feature selection for neural network based detection of flaws in steel welded joints using ultrasound testing. *Ultrasonics* 73:1–8
40. Li J, Qi C, Li Y, Wu Z (2021) Prediction and compensation of contour error of CNC systems based on LSTM neural-network. *IEEE/ASME Trans Mechatron*, 1–1
41. Hu C, Ou T, Chang H, Zhu Y, Zhu L (2020) Deep GRU neural network prediction and feedforward compensation for precision multi-axis motion control systems. *IEEE/ASME Trans Mechatron* 25(3):1377–1388
42. Erb RJ (1993) Introduction to backpropagation neural network computation. *Pharm Res* 10(2):165–170
43. Er MJ, Wu S, Lu J, Toh HL (2002) Face recognition with radial basis function (RBF) neural networks. *IEEE Trans Neural Netw* 13(3):697–710
44. Park J, Sandberg IW (1991) Universal approximation using radial-basis-function networks. *Neural Comput* 3(2):246–257
45. Gao Z (2006) Scaling and bandwidth-parameterization based controller tuning. In: Proceedings of the American control conference 6:4989–4996
46. Scheel M, Berndt A, Simanski O (2015) Iterative learning control: an example for mechanical ventilated patients. *IFAC-PapersOnLine* 48(20):523–527

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.