Hermite Broad-Learning Recurrent Neural Control with Adaptive Learning Rate for Nonlinear Systems

Chun-Fei Hsu (140930@o365.tku.edu.tw)
Tamkang University https://orcid.org/0000-0002-1950-8774
Bo-Rui Chen
National Yang Ming Chiao Tung University

Research Article

Keywords: Hermite broad-learning system, online parameter learning, adaptive learning rate, Lyapunov function, gradient descent approach

Posted Date: August 28th, 2023

DOI: https://doi.org/10.21203/rs.3.rs-3133875/v1

Version of Record: A version of this preprint was published at Soft Computing on December 16th, 2023.
See the published version at https://doi.org/10.1007/s00500-023-09481-2.

Hermite Broad-Learning Recurrent Neural Control with Adaptive Learning Rate for Nonlinear Systems

Chun-Fei Hsu[1] and Bo-Rui Chen[2]
[1] Department of Electrical Engineering, Tamkang University,
New Taipei City 25137, Taiwan
E-mail: fei@ee.tku.edu.tw
[2] Department of Electrical and Computer Engineering, Institute of Electrical and Control Engineering, National Yang Ming Chiao Tung University, Hsinchu 30010, Taiwan
E-mail: gogo40101.ee08@nycu.edu.tw

Abstract: Although conventional control systems are simple and widely used, they may not be effective for complex and uncertain systems. This study proposes a Hermite broad-learning recurrent neural network (HBRNN) to address such challenges. The HBRNN has a wide network structure and incorporates an internal feedback loop that enables fast learning and dynamic mapping. Furthermore, a Hermite broad-learning recurrent neural control (HBRNC) with the HBRNN as the main controller is proposed. All the network parameters of the HBRNN are updated online according to parameter learning laws through the gradient descent approach. To prevent network parameter overtraining of the HBRNN, an adaptive learning rate (ALR) is established using a discrete-type Lyapunov function to determine the least upper bound for the learning rate. The ALR can dynamically adjust the learning rates within specified ranges during the training process, thus achieving an appropriate balance between convergence speed and system stability. Finally, the HBRNC system with ALR is applied to a chaotic circuit and a reaction wheel pendulum, and its effectiveness is validated through simulation and experimentation.

Keywords: Hermite broad-learning system; online parameter learning; adaptive learning rate; Lyapunov function; gradient descent approach.

1. Introduction

In linear systems, the output is directly proportional to the input, resulting in linear characteristics with a simple mathematical equation. Nonlinear systems can be found in various real-world applications in fields ranging from physics and engineering to biology, economics, and social sciences. Because of their complex and unpredictable behavior, nonlinear systems are unable to exhibit a linear relationship between the input and output variables, making their analysis and control more challenging compared to linear systems. To address these challenges, researchers have developed several methods for controlling nonlinear systems, including feedback linearization, sliding-mode control, adaptive control, and backstepping control, among others. These control systems rely on mathematical models of system dynamics; however, obtaining accurate models for many real-time control applications can be challenging or even impossible.

Recently, deep-learning neural networks have emerged as powerful tools for modeling and approximating complex nonlinear relationships, even in changing or uncertain environments, making them well-suited for solving control problems that lack explicit knowledge of dynamic models (Cheng et al., 2019; Fang et al., 2019; Elhaki and Shojaei, 2020; Huynh et al., 2020; Zhang et al., 2022; Zhang and Wai, 2022). However, deep-learning neural networks have certain drawbacks, including complex network structures, high computational power requirements, and the requirement of a substantial amount of training data. These complexities pose challenges for deploying and running deep-learning neural networks in real-time control applications, where quick and efficient processing is crucial.

As an alternative learning architecture, a broad learning system (BLS) was proposed to address some of the limitations of deep-learning neural networks (Chen and Liu, 2018). Unlike deep-learning neural networks, the BLS is realized through broad expansion in order to reduce the computational complexity, memory requirements, and processing delays associated with

deep-learning neural networks (Wang et al., 2021; Tian et al., 2022; Yi et al., 2022). The BLS achieves good performance without requiring a large number of training parameters, making it more suitable for real-time control applications. Feng and Chen (2018) proposed a BLS control system in which network parameters are tuned online using the gradient descent approach; however, the control strategy has not yet undergone a stability analysis. Sui et al. (2020) and Huang et al. (2020) utilized a BLS to identify unknown dynamic models while ensuring global stability and finite-time stability. Xu et al. (2022) employed a BLS to develop a control strategy for microswimmer trajectory tracking. Additionally, Yuan et al. (2022) applied approximation-based adaptive optimal control and a BLS to model unknown dynamics and approximated an optimal cost function and an optimal control law using two BLSs.

Although these BLS controllers outperform deep-learning neural controllers in terms of robustness and convergence speed, the sparse autoencoder in the BLS requires numerous iterations to converge to an optimal solution. To mitigate the computational burden, a fuzzy broad learning system (FBLS) was proposed, where each feature node is replaced by a set of fuzzy logic systems (Tsai et al., 2020; Han et al., 2022; Bai et al., 2022; Chen et al., 2022). Although the FBLS demonstrates good effectiveness and low computational cost, a major drawback is that its inherent feed-forward network structure limits its application to static problems. To enable dynamic mapping capabilities, Du et al. (2021), Huang et al. (2021) and Wang et al. (2022) proposed a broad long short-term memory network and Hsu et al. (2022) proposed a broad-learning recurrent Hermite neural network to enhance the learning process for modeling complex systems while maintaining the same width design as the BLS and FBLS.

However, determining an appropriate learning rate in parameter learning laws can be a challenging and time-consuming task. If the learning rate is too law, then convergence may require an excessively long time, whereas if the learning rate is too high, this can lead to oscillations during the learning process. In most studies, the learning rate is typically set as a

constant and determines the step size for each parameter update. However, employing a fixed learning rate can sometimes result in suboptimal convergence or slow learning (Feng and Chen, 2018; Sui et al., 2020; Huang et al., 2020; Xu et al., 2022; Yuan et al., 2022). Instead of using a fixed learning rate, researchers have explored the use of adaptive learning rate (ALR) techniques to optimize learning performance. Fan et al. (2019) proposed a time-based learning rate schedule, where the learning rates decrease over time and iterations. Zhao and Lin (2019), Lin et al. (2021), and Tsai et al. (2020) determined the least upper bound of the learning rate through theoretical analysis using a discrete-type Lyapunov function to ensure system stability. Le and Ngo (2022) applied the Jaya algorithm to obtain the optimal learning rate for the designed parameter adaptation laws.

Motivated by the above discussion, this study proposes a Hermite broad-learning recurrent neural network (HBRNN), which has the advantages of high accuracy, significantly short training time, and dynamic mapping ability. Additionally, the proposed HBRNN can mimic an ideal controller and be used to realize a Hermite broad-learning recurrent neural control (HBRNC) system, which can be operated without explicit knowledge of dynamic models. The HBRNC system consists of a neural controller using an HBRNN and a supervisory controller to handle approximator errors introduced by the neural controller. The stability of the HBRNC system was ensured using a Lyapunov function. To enhance the learning capability of the HBRNN, a fully tuned parameter learning law was applied to adjust all the parameters of the HBRNN based on the online gradient descent method. Furthermore, this study proposes a novel ALR based on a discrete-type Lyapunov function, capable of dynamically adjusting the learning rate of parameter learning laws during the training process to ensure the convergence of tracking errors and increase the convergence speed. Finally, the HBRNC system was applied to a chaotic circuit and a reaction wheel pendulum. The simulation and experimental results indicated that the HBRNC system with ALR can achieve superior control performance.

2. Problem Formulation

Consider an *n*th-order nonlinear system defined as follows:

$$x^{(n)} = f(\mathbf{x}) + g(\mathbf{x})u \tag{1}$$

where $\mathbf{x} = [x, \dot{x}, \ldots, x^{(n-1)}]^T$ is the state vector, $f(\mathbf{x})$ and $g(\mathbf{x}) \neq 0$ represent the system dynamics, and is the control input. A control rule is designed such that the system state x closely tracks a desired reference command x_c. The tracking error can be defined as:

$$e = x_c - x. \tag{2}$$

To simplify the controller design process, the error dynamics can be expressed as follows (Hsu and Lee, 2017):

$$e^{(n)} = z(\mathbf{x}) - u \tag{3}$$

where:

$$z(\mathbf{x}) = x_c^{(n)} - (1 - \frac{1}{g(\mathbf{x})})x^{(n)} - \frac{f(\mathbf{x})}{g(\mathbf{x})} \tag{4}$$

is the lumped dynamics term. Therefore, an ideal controller u^* can be designed as (Slotine and Li, 1991):

$$u^* = z(\mathbf{x}) + \mathbf{k}^T\mathbf{e} \tag{5}$$

where $\mathbf{k} = [k_1, k_2, ..., k_n]^T$, $\mathbf{e} = [e, \dot{e}, ..., e^{(n-1)}]^T$, and $k_i, i = 1,2,..,n$ are positive constant gains. Substituting (5) into (3) yields:

$$e^{(n)} + \mathbf{k}^T\mathbf{e} = 0. \tag{6}$$

The dynamics equation (6) can be expressed as a state vector as follows:

$$\dot{\mathbf{e}} = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ 0 & 0 & 1 & \cdots \\ \vdots & \vdots & \vdots & 1 \\ -k_1 & -k_2 & \cdots & -k_n \end{bmatrix}\mathbf{e} = \mathbf{\Lambda e} \tag{7}$$

where:

$$\mathbf{\Lambda} = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ 0 & 0 & 1 & \cdots \\ \vdots & \vdots & \vdots & 1 \\ -k_1 & -k_2 & \cdots & -k_n \end{bmatrix}. \tag{8}$$

Using appropriate k_i, $i = 1,2,..,n$, the control performances such as rise time, settling time, and overshoot can be determined (Slotine and Li, 1991). However, designing accurate models with the lumped dynamics term $z(\mathbf{x})$ can be challenging or even impossible in many real-time control applications. On the other hand, sliding-mode control is a robust and effective control technique with various advantages, such as robustness, fast response, and insensitivity to modeling errors, making it a valuable approach for various control applications (Slotine and Li, 1991). A sliding surface with a special integral term was proposed by Wai et al. (2014):

$$s = C(\mathbf{e}) - C(\mathbf{e}_0) - \int_0^t \frac{\partial C(\mathbf{e})}{\partial \mathbf{e}} \mathbf{\Lambda} \mathbf{e} d\tau \tag{9}$$

where $C(\mathbf{e})$ is the vector to be designed, and $\mathbf{e}_0$ is the initial state of $\mathbf{e}$. From (8) and (9), $s = 0$ at $t = 0$ and:

$$\dot{s} = \frac{\partial C(\mathbf{e})}{\partial \mathbf{e}} \dot{\mathbf{e}} - \frac{\partial C(\mathbf{e})}{\partial \mathbf{e}} \mathbf{\Lambda} \mathbf{e} = 0. \tag{10}$$

Unlike traditional sliding-mode control, the sliding surface is maintained at at all time instants. There is also no reaching phase, so the ideal controller provides superior robustness against system uncertainties and disturbances (Slotine and Li, 1991).

3. Description of HBRNN

Although deep-learning neural networks have achieved remarkable success in various control applications, they often consist of multiple layers with a large number of interconnected nodes or neurons. As a result, deep-learning neural networks have high computational and

memory requirements, making them unsuitable for real-time control applications that necessitate fast and efficient processing. Figure 1 illustrates the network structure of the proposed HBRNN, which comprises an input layer, a recurrent feature layer, an enhancement layer, and an output layer.

Layer 1 (Input Layer):

This is the first layer of HBRNN, where the input data is fed into the network. The input variable to layer 1 is given by the sliding surface s in (9) to reduce the network complexity and computational burden.

Layer 2 (Recurrent Feature Layer):

The recurrent feature layer is responsible for capturing temporal dependencies in the input sequence. There are m feature subsystems in HBRNN, where the *i*-th Hermite node of the *j*-th feature subsystem is given as (Hsu et al., 2022):

$$h_{ji}(s) = \frac{1}{\sqrt{2^i i!\sqrt{\pi}}} e^{-(s^2/2)} H_i(s), \text{ for } i = 1,2,...,l \tag{11}$$

where:

$$\begin{aligned} &H_1(s) = 1 \\ &H_2(s) = 2s \\ &\vdots \\ &H_l(s) = 2sH_{l-1}(s) - 2(l-1)H_{l-2}(s),\ l \geq 3 \end{aligned} \tag{12}$$

The output of *j*-th feature subsystem can be expressed as:

$$V_j = \sum_{i=0}^{l} v_{ji} h_{ji}, \text{ for } j = 1,2,...,m \tag{13}$$

where v_{ji} are the adjustable parameters of *j*-th subsystem and the output signal of recurrent node is given as:

$$h_{j0} = \psi(V_j^{pre} + \kappa \cdot h_{j0}^{pre}) \tag{14}$$

in which V_j^{pre} denotes the output signal of *j*-th subsystem in the previous time, h_{j0}^{pre} denotes the output signal of a recurrent node in the previous time, $\psi(\cdot)$ is the sigmoid function, and $0 \le \kappa \le 1$ is the fixed feedback gain value. This feedback mechanism facilitates dynamic mapping by incorporating previous outputs as part of the current input, making HBRNN suitable for controlling nonlinear dynamic systems.

Layer 3 (Enhancement Layer):

The enhancement layer aims to enhance the representation learned from the recurrent feature layer by incorporating additional information or modeling higher-level dependencies. There are n groups of enhancement nodes in HBRNN. The output of *k*-th enhancement node can be expressed as:

$$W_k = \xi(\sum_{j=0}^{m} w_{jk} V_j), \text{ for } k = 1,2,...,n \tag{15}$$

where w_{jk} are the adjustable parameters of *k*-th group of enhancement nodes, $V_0 = 1$ and $\xi(\cdot)$ is the hyperbolic tangent function.

Layer 4 (Output Layer):

The output layer is the final layer of HBRNN, where the network produces the desired output. By connecting the Hermite feature layer and the enhancement layer, it means that HBRNN has a flatted structure. The HBRNN output denotes as:

$$u_{nc} = \sum_{j=1}^{m} \alpha_j V_j + \sum_{k=1}^{n} \beta_k W_k \tag{16}$$

where α_j and β_k are the adjustable parameters of output layer. For ease of notation, the HBRNN output can be re-expressed as:

$$u_{nc} = \boldsymbol{\alpha}^T \mathbf{V} + \boldsymbol{\beta}^T \mathbf{W} \tag{17}$$

where $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_m]^T$, $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_n]^T$, $\mathbf{V} = [V_1, \ldots, V_m]^T$ and $\mathbf{W} = [W_1, \ldots, W_n]^T$. Assume that an optimal HBRNN can approximate the ideal controller u^* in (5) such that (Wnag, 1994):

$$u^* = \boldsymbol{\alpha}^{*T}\mathbf{V} + \boldsymbol{\beta}^{*T}\mathbf{W} + \varepsilon \tag{18}$$

where $\boldsymbol{\alpha}^*$ and $\boldsymbol{\beta}^*$ are the optimal parameter vectors of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, respectively, and ε denotes the approximation error assumed to be bounded as $0 \le |\varepsilon| \le E$.

It is worth noting that the HBRNN incorporates an internal feedback loop to capture the dynamic responses without relying on external feedback through delays, thus improving the dynamic characteristics and convergence speed. Unlike traditional deep-learning neural networks that rely on stacking multiple layers to capture complex representations, the proposed HBRNN extends the network width while maintaining a shallow architecture. Thus, the HBRNN has fewer hyperparameters compared with deep-learning neural networks, making it more suitable for controller design applications.

4. Design of the HBRNC System

The proposed HBRNC system for the *n*th-order nonlinear system, as shown in Fig. 2, can be defined as:

$$u_{hc} = u_{nc} + u_{sc} = \hat{\boldsymbol{\alpha}}^T\mathbf{V} + \hat{\boldsymbol{\beta}}^T\mathbf{W} + u_{sc} \tag{19}$$

The neural controller u_{nc} uses an HBRNN to approximate the ideal controller u^* in (5), the supervisor controller u_{sc} is designed to cope with the approximator error introduced by the neural controller, and $\hat{\boldsymbol{\alpha}}$ and $\hat{\boldsymbol{\beta}}$ are the optimal parameter vectors of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, respectively.

4.1. Stability Analysis

Substituting (19) into (3) and using (5) yield:

$$\dot{\mathbf{e}} = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ 0 & 0 & 1 & \cdots \\ \vdots & \vdots & \vdots & 1 \\ -k_1 & -k_2 & \cdots & -k_n \end{bmatrix} \mathbf{e} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} (u^* - u_{nc} - u_{sc})$$

$$= \mathbf{\Lambda e} + \mathbf{b}(u^* - u_{nc} - u_{sc}) \tag{20}$$

where $\mathbf{b} = [0,0,...,1]^T$. A sliding surface is choose as (9), then differentiating (9) with respect to time and using (20) give as:

$$\dot{s} = \frac{\partial C(\mathbf{e})}{\partial \mathbf{e}} \dot{\mathbf{e}} - \frac{\partial C(\mathbf{e})}{\partial \mathbf{e}} \mathbf{\Lambda e}$$

$$= u^* - u_{nc} - u_{sc} \tag{21}$$

where $C(\mathbf{e})$ is designed to satisfy $\frac{\partial C(\mathbf{e})}{\partial \mathbf{e}} = [0,0,...,1]$. Using the universal approximation theorem (Wnag, 1994), (21) can be rewritten as:

$$\dot{s} = \boldsymbol{\alpha}^{*T} \mathbf{V} + \boldsymbol{\beta}^{*T} \mathbf{W} + \varepsilon - \hat{\boldsymbol{\alpha}}^T \mathbf{V} - \hat{\boldsymbol{\beta}}^T \mathbf{W} - u_{sc}$$

$$= \tilde{\boldsymbol{\alpha}}^T \mathbf{V} + \tilde{\boldsymbol{\beta}}^T \mathbf{W} + \varepsilon - u_{sc} \tag{22}$$

where $\tilde{\boldsymbol{\alpha}} = \boldsymbol{\alpha}^* - \hat{\boldsymbol{\alpha}}$ and $\tilde{\boldsymbol{\beta}} = \boldsymbol{\beta}^* - \hat{\boldsymbol{\beta}}$. To analysis the stability of the HBRNC system, a Lyapunov function is defined by:

$$V_1 = \frac{1}{2} s^2 + \frac{1}{2\eta_\alpha} \tilde{\boldsymbol{\alpha}}^T \tilde{\boldsymbol{\alpha}} + \frac{1}{2\eta_\beta} \tilde{\boldsymbol{\beta}}^T \tilde{\boldsymbol{\beta}} \tag{23}$$

where η_α and η_β are the learning rates. Taking the derivative of the Lyapunov function yields:

$$\dot{V}_1 = s\dot{s} + \frac{1}{\eta_\alpha} \tilde{\boldsymbol{\alpha}}^T \dot{\tilde{\boldsymbol{\alpha}}} + \frac{1}{\eta_\beta} \tilde{\boldsymbol{\beta}}^T \dot{\tilde{\boldsymbol{\beta}}}$$

$$= s(\tilde{\boldsymbol{\alpha}}^T \mathbf{V} + \tilde{\boldsymbol{\beta}}^T \mathbf{W} + \varepsilon - u_{sc}) - \frac{1}{\eta_\alpha} \tilde{\boldsymbol{\alpha}}^T \dot{\hat{\boldsymbol{\alpha}}} - \frac{1}{\eta_\beta} \tilde{\boldsymbol{\beta}}^T \dot{\hat{\boldsymbol{\beta}}}$$

$$= \tilde{\boldsymbol{\alpha}}^T (s\mathbf{V} - \frac{1}{\eta_\alpha} \dot{\hat{\boldsymbol{\alpha}}}) + \tilde{\boldsymbol{\beta}}^T (s\mathbf{W} - \frac{1}{\eta_\beta} \dot{\hat{\boldsymbol{\beta}}}) + s(\varepsilon - u_{sc}) \tag{24}$$

If the parameter learning laws and supervisor controller are derived as:

$$\dot{\hat{\boldsymbol{\alpha}}} = \eta_{\alpha} s\mathbf{V} \tag{25}$$

$$\dot{\hat{\boldsymbol{\beta}}} = \eta_{\beta} s\mathbf{W} \tag{26}$$

$$u_{sc} = E\,\mathrm{sgn}(s) \tag{27}$$

(24) can be obtained as:

$$\begin{aligned} \dot{V}_1 &= \varepsilon s - E|s| \\ &\le |\varepsilon||s| - E|s| \\ &= -(E - |\varepsilon|)|s| \le 0\,. \end{aligned} \tag{28}$$

Therefore, the stability of the HBRNC system can be guaranteed using the parameter learning laws (25) and (26) and supervisor controller (27) in the sense of the Lyapunov function (Slotine and Li, 1991).

4.2. Full-Tuned Parameter Learning Law

To increase the learning capability of HBRNN, a full-tuned parameter learning law should be considered to online adjust all the parameters of HBRNN. A cost function is defined as:

$$\Omega = \frac{1}{2} e^2\,. \tag{29}$$

The parameter learning law of parameters α_j in output layer using the gradient descent method can be represented as (Lin and Lee, 1996):

$$\begin{aligned} \Delta\alpha_j &= -\eta_{\alpha} \frac{\partial \Omega}{\partial \alpha_j} \\ &= -\eta_{\alpha} \frac{\partial \Omega}{\partial u_{nc}} \frac{\partial u_{nc}}{\partial \alpha_j} \\ &= -\eta_{\alpha} \frac{\partial \Omega}{\partial u_{nc}} V_j \end{aligned} \tag{30}$$

where $\frac{\partial \Omega}{\partial u_{nc}}$ is the Jacobian term of the system. Comparing (25) with (30), the Jacobian term can be obtained as:

$$\frac{\partial \Omega}{\partial u_{nc}} = -s \tag{31}$$

and

$$\frac{\partial e}{\partial u_{nc}} = -\frac{s}{e}. \tag{32}$$

Thus, the parameter learning law of parameters α_j and β_k can be represented as:

$$\Delta \alpha_j = \eta_\alpha s V_j \tag{33}$$

$$\begin{aligned} \Delta \beta_k &= -\eta_\beta \frac{\partial \Omega}{\partial \beta_k} \\ &= -\eta_\beta \frac{\partial \Omega}{\partial u_{nc}} \frac{\partial u_{nc}}{\partial \beta_j} \\ &= \eta_\beta s W_k \end{aligned} \tag{34}$$

Further, the parameter learning laws of parameters w_{jk} and v_{ji} can be given as:

$$\begin{aligned} \Delta w_{jk} &= -\eta_w \frac{\partial \Omega}{\partial w_{jk}} \\ &= -\eta_w \frac{\partial \Omega}{\partial u_{nc}} \frac{\partial u_{nc}}{\partial W_k} \frac{\partial W_k}{\partial w_{jk}} \\ &= \eta_w s \beta_k \left(1 - \xi^2 (\sum_{j=0}^{m} w_{jk} V_j) \right) V_j \\ &= \eta_w s \beta_k \left(1 - W_k^2 \right) V_j \end{aligned} \tag{35}$$

$$\Delta v_{ji} = -\eta_v \frac{\partial \Omega}{\partial v_{ji}}$$

$$= -\eta_v \frac{\partial \Omega}{\partial u_{nc}} \left(\frac{\partial u_{nc}}{\partial V_j} + \sum_{k=1}^{n} \frac{\partial u_{nc}}{\partial W_k} \frac{\partial W_k}{\partial V_j} \right) \frac{\partial V_j}{\partial v_{ji}}$$

$$= \eta_v s \left(\alpha_j + \sum_{k=1}^{n} \beta_k w_{jk} \left(1 - \xi^2 \left(\sum_{j=0}^{m} w_{jk} V_j \right) \right) \right) h_{ji}$$

$$= \eta_v s \left(\alpha_j + \sum_{k=1}^{n} \beta_k w_{jk} \left(1 - W_k^2 \right) \right) h_{ji} \tag{36}$$

where η_w and η_v are the learning rates.

4.3 Adaptive Learning Rate

The selection of learning rates in the parameter learning laws (33)–(36) has a significant impact on the network performance of HBRNN. If the learning rates are too high, the network may undergo oscillations or fail to converge during the learning process. Conversely, if the leaning rates are too low, then this may result in slow parameter learning. In this study, an ALR was derived for the designed parameter learning laws to accelerate the convergence of HBRNN parameters. The ALR assigns four specific learning rates to the designed parameter learning laws (33)–(36) to ensure accurate tracking control of the HBRNC system. The learning rates for each layer are specified in the Appendix.

Unlike the previous study conducted by Hsu et al. (2022), this study incorporated two features. (1) A sliding surface with a special integral term was employed through a sliding-mode control approach to enhance robustness and the complexity of the network was further reduced by feeding the sliding surface into an HBRNN. (2) A discrete-type Lyapunov function was employed to determine the learning rate ranges, ensuring the convergence of the tracking error. Additionally, the ALR was applied to adjust the learning rates within a given range to accelerate the convergence of network parameters.

5. Simulation and Experimental Results

The effectiveness of the proposed HBRNC system with ALR was verified using two systems: a chaotic circuit and a reaction wheel pendulum.

5.1. Chaotic Circuit

The chaotic circuit is a well-known electronic circuit that exhibits chaotic behavior. It is designed to produce complex and nonlinear dynamics, including chaos and bifurcations. The behavior of a chaotic circuit is highly sensitive to initial conditions and system parameters. It has been utilized in various applications, such as random number generators, secure communications, and even artistic installations. The dynamic equation of the chaotic circuit, as proposed by Wang et al. (2002), is expressed as follows:

$$x^{(3)} = f(\mathbf{x}) + u \tag{37}$$

where $\mathbf{x} = [x, \dot{x}, \ddot{x}]^T$ is the state vector,

$$f(\mathbf{x}) = \frac{14}{1805}x - \frac{168}{9025}\dot{x} + \frac{1}{38}\ddot{x} - \frac{2}{45}\left(\frac{28}{361}x + \frac{7}{95}\dot{x} + \ddot{x}\right)^3 \tag{38}$$

is the system dynamics, and u is the control input. The open-loop system behavior is simulated with two different initial points, as shown in Fig. 3. The uncontrolled chaotic system is sensitive to the initial points, such that even small changes in the initial points can lead to significantly different trajectories over time.

To demonstrate the control performance of the HBRNC system in this study, a set of low learning rates ($\eta_\alpha = \eta_\beta = \eta_w = \eta_v = 0.01$) was applied to control the chaotic circuit. The controller parameters were set as $k_1 = 1$, $k_2 = 3$, $k_3 = 3$, $\kappa = 0.5$, and $E = 0.01$. The simulation results of the HBRNC system with low learning rates for the chaotic circuit are shown in Fig. 4. They show that the HBRNC system with low learning rates can easily achieve parameter convergence; however, the corresponding learning speed is low. Subsequently, a set of high learning rates ($\eta_\alpha = \eta_\beta = \eta_w = \eta_v = 0.8$) was employed to accelerate the convergence

of the tracking error. The simulation results of the HBRNC system with high learning rates for the chaotic circuit are shown in Fig. 5. They show that the HBRNC system with high learning rates can achieve better control performance and faster convergence; however, the parameter overtraining of the HBRNN caused a phenomenon called chattering. Comparing Figs. 4 and 5 reveals that a fixed learning rate may not be optimal for all stages of training.

Finally, the HBRNC system with ALR was employed to control the chaotic circuit again, where the learning rates (η_α, η_β, η_w and η_v) were dynamically adjusted during the training process using the developed ALR. The simulation results of the HBRNC system with ALR for the chaotic circuit are shown in Fig. 6. They show that the HBRNC system with ALR can achieve faster convergence and better performance for both initial points compared to the HBRNC system with a fixed learning rate.

5.2. Reaction Wheel Pendulum

As shown in Fig. 7, the reaction wheel pendulum offers a challenging and practical nonlinear system for testing and verifying control schemes. The reaction wheel is a spinning flywheel that can generate a torque when its angular momentum is changed. The dynamic equation of the reaction wheel pendulum is given as (Zhang et al., 2020; Jiang and Astolfi, 2021; Chen et al., 2022):

$$\ddot{\phi} = -\frac{(m_p l_{cp} + m_r l_p) g}{m_p l_{cp}^2 + m_r l_p^2 + J_p} \sin\phi - \frac{2}{m_p l_{cp}^2 + m_r l_p^2 + J_p} \tau \tag{39}$$

where ϕ is the pendulum angle, τ is the control input torque acting on the reaction wheel, and definitions of other parameter symbols are given in Table 1. The control objective is to drive the pendulum at an upward unstable balance point and keep it controlled there. To investigate the effectiveness of the proposed HBRNC method, a payload scenario and a disturbance scenario are tested. A comparison among the fuzzy neural control (FNC) (Chang et al., 2018),

the fuzzy broad-learning neural control (FBNC) (Chen et al., 2022), and the proposed HBRNC is made.

First, the FNC system (Chang et al., 2018) is applied to the reaction wheel pendulum. The structure settings of the used fuzzy neural network are as follows: one node in the input layer, seven Gaussian functions in the membership layer, and one node in the output layer. The controller parameters are selected as $k_1 = 20$, $k_2 = 100$, $\kappa = 0.2$, $\eta_\omega = 0.1$, and $\eta_\sigma = \eta_c = 0.01$, where η_ω is the learning rate in the output layer, and η_σ and η_c are the learning rates in the membership layer. The experimental results of the AFNC system for the reaction wheel pendulum are shown in Fig. 8. They demonstrate that the AFNC system can stabilize the pendulum at the upward unstable equilibrium point, under the effect of the added payloads and external disturbances. However, the swing amplitude on the pendulum is large in both test scenarios.

Next, the FBNC system (Chen et al., 2022) is applied to the reaction wheel pendulum again. The structure settings of the used FBLS are as follows: five fuzzy subsystems, three fuzzy sets of each input, two enhancement nodes, and one output node. The controller parameters are selected as $k_w = 0.4$, $k_1 = 20$, $k_2 = 100$, $\eta_\alpha = \eta_\beta = 0.1$, $\eta_\sigma = \eta_c = 0.01$, and $\eta_a = \eta_b = 0.001$, where η_α and η_β are the learning rates in the output layer, η_σ and η_c are the learning rates in the feature layer, and η_a and η_b are the learning rates in the enhancement layer. The experimental results of the AFBNC system for the reaction wheel pendulum are shown in Fig. 9. It shows that the AFBNC system can stabilize the pendulum at the upward unstable equilibrium point with a smaller swing amplitude and faster convergence speed, thanks to the stronger capability of the FBLS, compared to the fuzzy neural network.

Finally, the HBRNC system is applied to the reaction wheel pendulum again. The structure settings of the used HBRNN are as follows: five feature subsystems, three Hermite nodes for

each input, and three enhancement nodes. The controller parameters are selected as $k_1 = 20$, $k_2 = 100$, and $\kappa = 0.2$. The experimental results of the HBRNC system with low learning rates ($\eta_\alpha = \eta_\beta = 0.02$, $\eta_w = \eta_v = 0.01$) for the reaction wheel pendulum are shown in Fig. 10. Though the HBRNC system with low learning rates can stabilize the pendulum at the upward unstable equilibrium point, the corresponding learning speed is low. To increase the convergence speed, the experimental results of the HBRNC system with high learning rates ($\eta_\alpha = \eta_\beta = \eta_w = \eta_v = 0.1$) for the reaction wheel pendulum are shown in Fig. 11. It demonstrates that better control performance and faster convergence can be obtained. However, in both test scenarios, it is observed that the chattering phenomenon of control input results in high-frequency and small jitter in the pendulum angle, which is caused by the parameter overtraining problem of HBRNN. The chattering phenomenon will result in damage to actuators or plants, and the system may even eventually become unstable. Setting an appropriate learning rate is crucial as it affects the convergence speed of the network parameters and the outcome of the control response.

Additionally, the HBRNC system with ALR is applied to the reaction wheel pendulum again. The controller parameters are selected as $k_1 = 20$, $k_2 = 100$, and $\kappa = 0.2$. The learning rates (η_α, η_β, η_w and η_v) are dynamically adjusted during the training process using the proposed ALR. The experimental results of the HBRNC system with ALR for the reaction wheel pendulum are shown in Fig. 12. They show the HBRNC system with ALR in both test scenarios not only successfully stabilizes the pendulum at the upward unstable equilibrium point but also effectively mitigates the chattering phenomenon caused by the parameter overtraining problem. Utilizing the developed ALR, the HBRNC system can ensure that the reaction wheel pendulum system has a fast tracking error convergence speed and can prevent

the control signal from oscillating to help maintain the stability of the reaction wheel pendulum system.

5 Conclusions

The main contributions of this study are as follows. (1) A Hermite broad-learning recurrent neural network (HBRNN) was designed to facilitate rapid learning and dynamic mapping by employing a specific network structure constructed through broad expansion, coupled with an internal feedback loop. (2) A Hermite broad-learning recurrent neural control (HBRNC) system was developed for controlling an unknown nonlinear system. The parameters of the proposed HBRNN can be updated online using the designed parameter learning laws, and the stability of the HBRNC system can be ensured through Lyapunov stability analysis. (3) A discrete-type Lyapunov function was employed to determine the learning rate ranges that can ensure the convergence of tracking error. An adaptive learning rate (ALR) was employed to adjust the learning rate within a specified range in order to accelerate the convergence of network parameters. (4) Simulation and experimental results showed that the proposed HBRNC system with ALR is effective in handling control problems. Finally, a comparison of control characteristics, presented in Table 2, showed that the HBRNC system with ALR not only has the advantages of simplicity and lightweight implementation but also the ability to quickly adapt network parameters and avoid overtraining problems.

Appendix

Theorem 1: Let η_α, η_β, η_w and η_v be the learning rates for the parameters learning laws of HBRNN. Define $P_{\alpha\max}$ as $P_{\alpha\max} = \max_N \left\| P_\alpha(N) \right\|$, where $P_\alpha(N) = \dfrac{\partial \tau_{nc}}{\partial \alpha_j}$; define $P_{\beta\max}$ as

$P_{\beta\max} = \max_N \left\| P_\beta(N) \right\|$, where $P_\beta(N) = \dfrac{\partial u_{nc}}{\partial \beta_k}$; define $P_{w\max}$ as $P_{w\max} = \max_N \left\| P_w(N) \right\|$, where $P_w(N) = \dfrac{\partial \tau_{nc}}{\partial w_{jk}}$; define $P_{v\max}$ as $P_{v\max} = \max_N \left\| P_v(N) \right\|$, where $P_v(N) = \dfrac{\partial u_{nc}}{\partial v_{ji}}$. Thus, the system stability can be guaranteed if η_α and η_β are chosen as $\eta_\alpha^* = \dfrac{1}{m(V_{\max})^2}$ and $\eta_\beta^* = \dfrac{2}{n}$, respectively, in which $V_{\max} = \max_j \left| V_j \right|$; η_w and η_v are chosen as $\eta_w^* = \dfrac{2}{mn(\beta_{\max} V_{\max})^2}$ and $\eta_v^* = \dfrac{2}{n(\alpha_{\max} + \beta_{\max} w_{\max})^2}$, respectively, in which $\alpha_{\max} = \max_j \left| \alpha_j \right|$, $\beta_{\max} = \max_k \left| \beta_k \right|$ and $w_{\max} = \max_{j,k} \left| w_{jk} \right|$.

Proof: Since $P_\alpha(N) = \dfrac{\partial \tau_{nc}}{\partial \alpha_j} = V_j$ and $P_\beta(N) = \dfrac{\partial u_{nc}}{\partial \beta_k} = W_k$, the following result can be concluded:

$$\left\| P_\alpha(N) \right\| < \sqrt{m} V_{\max} \tag{A1}$$

$$\left\| P_\beta(N) \right\| < \sqrt{n} \tag{A2}$$

The upper bounds of $P_w(N)$ and $P_v(N)$ can be derived as follows:

$$\begin{aligned} P_w(N) &= \frac{\partial u_{nc}}{\partial w_{jk}} = \frac{\partial u_{nc}}{\partial W_k} \frac{\partial W_k}{\partial w_{jk}} \\ &= \beta_k \left(1 - W_k^2\right) V_j \\ &\le \beta_k V_j \end{aligned} \tag{A3}$$

$$\begin{aligned} P_v(N) &= \frac{\partial u_{nc}}{\partial v_{ji}} = \left(\frac{\partial u_{nc}}{\partial V_j} + \sum_{k=1}^{n} \frac{\partial u_{nc}}{\partial W_k} \frac{\partial W_k}{\partial V_j} \right) \frac{\partial V_j}{\partial v_{ji}} \\ &= \left(\alpha_j + \sum_{k=1}^{n} \beta_k w_{jk} \left(1 - W_k^2\right) \right) h_{ji} \end{aligned}$$

$$\le \alpha_j + \sum_{k=1}^{n} \beta_k w_{jk} \left(1 - W_k^2\right)$$

$$\le \alpha_j + \sum_{k=1}^{n} \left|\beta_k\right| \left|w_{jk}\right| \tag{A4}$$

From (A3) and (A4), the inequalities can be obtained as:

$$\left\|P_w(N)\right\| \le \left\|\beta_k V_j\right\| \le \left\|\beta_k\right\| \left\|V_j\right\| \le \sqrt{mn}\,\beta_{\max} V_{\max} \tag{A5}$$

$$\left\|P_v(N)\right\| \le \left\|\alpha_j + \sum_{k=1}^{n} \left|\beta_k\right| \left|w_{jk}\right|\right\|$$

$$\le \left\|\alpha_j\right\| + \left\|\sum_{k=1}^{n} \left|\beta_k\right| \left|w_{jk}\right|\right\|$$

$$\le \alpha_{\max} + \sqrt{n}\,\beta_{\max} w_{\max} \tag{A6}$$

To ensure the system stability, consider a discrete-type Lyapunov function as follows:

$$V_2(N) = \frac{1}{2} e^2(N) \tag{A7}$$

where N denotes the number of iteration. The change of discrete-type Lyapunov function can be expressed as:

$$\Delta V_2(N) = V_2(N+1) - V_2(N) = \frac{1}{2}\left[e^2(N+1) - e^2(N)\right] \tag{A8}$$

The error difference can be represented by:

$$e(N+1) = e(N) + \Delta e(N)$$

$$= e(N) + \left[\frac{\partial e(N)}{\partial \alpha_j}\right]^T \Delta\alpha_j + \left[\frac{\partial e(N)}{\partial \beta_k}\right]^T \Delta\beta_k + \left[\frac{\partial e(N)}{\partial w_{jk}}\right]^T \Delta w_{jk} + \left[\frac{\partial e(N)}{\partial v_{ji}}\right]^T \Delta v_{ji} \tag{A9}$$

where $\Delta e(N)$ respects a change in system output, and $\Delta\alpha_j$, $\Delta\beta_k$, Δw_{jk}, and Δv_{ji} respect a parameter change in output layer, enhancement layer and recurrent feature layer, respectively.

Define $\xi = \dfrac{\partial e}{\partial u_{nc}}$ as a positive constant designed by the user, (A9) using (33)–(36), (A1), (A2), (A5) and (A6) can be obtained as:

$$\begin{aligned}\left\|e(N+1)\right\| &= \left\|e(N)\left(1-\eta_\alpha \xi^2 P_\alpha^T(N)P_\alpha(N)\right)+e(N)\left(1-\eta_\beta \xi^2 P_\beta^T(N)P_\beta(N)\right)\right. \\ &\quad \left.+e(N)\left(1-\eta_w \xi^2 P_w^T(N)P_w(N)\right)+e(N)\left(1-\eta_v \xi^2 P_v^T(N)P_v(N)\right)\right\| \\ &\le \left\|e(N)\right\|\left\|1-\eta_\alpha \xi^2 P_\alpha^T(N)P_\alpha(N)\right\|+\left\|e(N)\right\|\left\|1-\eta_\beta \xi^2 P_\beta^T(N)P_\beta(N)\right\| \\ &\quad +\left\|e(N)\right\|\left\|1-\eta_w \xi^2 P_w^T(N)P_w(N)\right\|+\left\|e(N)\right\|\left\|1-\eta_v \xi^2 P_v^T(N)P_v(N)\right\| \qquad \text{(A10)}\end{aligned}$$

If the learning rates for the parameters learning laws of HBRNN are selected as follows:

$$\eta_\alpha = \frac{1}{(\xi P_{\alpha\max})^2} = \frac{1}{m(\xi V_{\max})^2} \qquad \text{(A11)}$$

$$\eta_\beta = \frac{1}{(\xi P_{\beta\max})^2} = \frac{1}{n\xi^2} \qquad \text{(A12)}$$

$$\eta_w = \frac{1}{(\xi P_{w\max})^2} = \frac{1}{mn(\xi \beta_{\max} V_{\max})^2} \qquad \text{(A13)}$$

$$\eta_v = \frac{1}{(\xi P_{v\max})^2} = \frac{1}{(\xi(\alpha_{\max} + \sqrt{n}\beta_{\max} w_{\max}))^2} \qquad \text{(A14)}$$

the term $\left\|1-\eta_\alpha \delta^2 P_\alpha^T(N)P_\alpha(N)\right\|$, $\left\|1-\eta_\beta \delta^2 P_\beta^T(N)P_\beta(N)\right\|$, $\left\|1-\eta_w \delta^2 P_w^T(N)P_w(N)\right\|$ and $\left\|1-\eta_v \delta^2 P_v^T(N)P_v(N)\right\|$ are less than 1. According to $\left\|e(N+1)\right\| < \left\|e(N)\right\|$, the Lyapunov stability of $V_2(N) > 0$ and $\Delta V_2(N) < 0$ can be guaranteed. Thus, the discrete-type Lyapunov approach can find the learning rate ranges that allows the ALR can efficiently train HBRNN, where the learning rates are designed as $\eta_\alpha^* = \dfrac{\eta_\alpha}{2}$, $\eta_\beta^* = \dfrac{\eta_\beta}{2}$, $\eta_w^* = \dfrac{\eta_w}{2}$ and $\eta_v^* = \dfrac{\eta_v}{2}$ for the parameters learning laws of HBRNN (33)–(36), respectively.

Acknowledgment

This work was supported by the Ministry of Science and Technology (MOST), Taiwan, the Republic of China, under contract MOST 110-2221-E-032-038-MY2.

Declarations

Funding (The study was funded by the Ministry of Science and Technology of Republic of China under Grant MOST 110-2221-E-032-038-MY2)

Conflict of interest (We confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome)

Availability of data and material (Not applicable)

Code availability (Not applicable)

Data availability (Not applicable)

Authors' contributions (We confirm that the manuscript has been read and approved by all named authors and that there are no other persons who satisfied the criteria for authorship but are not listed)

Ethics approval (Not applicable)

Consent to participate (Not applicable)

Consent for publication (We confirm that there are no impediments to publication, including the timing of publication, with respect to intellectual property)

References

Bai, K., Zhu, X., Wen, S., Zhang, R., Zhang, W., 2022. Broad learning based dynamic fuzzy inference system with adaptive structure and interpretable fuzzy rules. IEEE Transactions on Fuzzy Systems 30(8), 3270–3283.

Chang, C.W., Hsu, C.F., Lee, T.T., 2018. Backstepping-based finite-time adaptive fuzzy control of unknown nonlinear systems. International Journal of Fuzzy Systems 20, 2545–2555.

Chen, B.R., Hsu, C.F., Wu. B.F., 2022. Microcontroller-based intelligent control for reaction wheel pendulums using a fuzzy broad-learning system. 2022 International Conference on Fuzzy Theory and Its Applications, 1–5.

Chen, C.L.P., Liu, Z., 2018. Broad learning system: an effective and efficient incremental learning system without the need for deep architecture. IEEE Transactions on Neural Networks and Learning Systems 29(1), 10–24.

Cheng, L., Wang, Z., Jiang, F., Zhou, C., 2019. Real-time optimal control for spacecraft orbit transfer via multiscale deep neural networks. IEEE Transactions on Aerospace and Electronic Systems 55(5), 2436–2450.

Du, J., Vong, C.M., Chen, C.L.P., 2021. Novel efficient RNN and LSTM-like architectures: recurrent and gated broad learning systems and their applications for text classification. IEEE Transactions on Cybernetics 51(3), 1586–1597.

Elhaki, O., Shojaei, K., 2020. A robust neural network approximation-based prescribed performance output-feedback controller for autonomous underwater vehicles with actuators saturation. Engineering Applications of Artificial Intelligence 88, 103382.

Fan, L., Zhang, T., Zhao, X., Wang, H., Zheng, M., 2019. Deep topology network: A framework based on feedback adjustment learning rate for image classification. Advanced Engineering Informatics 42, Art. no. 100935.

Fang, W., Chao, F., Yang, L., Lin, C.M., Shang, C., Zhou, C., Shen, Q., 2019. Arecurrent emotional cmac neural network controller for vision-based mobile robots. Neurocomputing 334, 227–238.

Feng, S., Chen, C.L.P., 2018. Broad learning system for control of nonlinear dynamic systems. 2018 IEEE International Conference on Systems, Man and Cybernetics, 2230–223.

Han, H.G., Liu, Z., Liu, H., Qiao, J., Chen, C.L.P., 2022. Type-2 fuzzy broad learning system. IEEE Transactions on Cybernetics 52(10), 10352–10363.

Hsu, C.F., Lee, T.T., 2017. Emotional fuzzy sliding-mode control for unknown nonlinear systems. International Journal of Fuzzy Systems 19, 942–953.

Hsu, C.F., Chen, B.R., Wu, B.F., 2022. Broad-learning recurrent Hermite neural control for unknown nonlinear systems. Knowledge-Based Systems 242, Art no. 108263.

Huang, H., Zhang, T., Yang, C., Chen, C.L.P., 2020. Motor learning and generalization using broad learning adaptive neural control. IEEE Transactions on Industrial Electronics 67(10), 8608–8617.

Huang, S., Rong, L., Chang, X., Wang, Z., Yuan, Z., Wei, C., Santos, O.J., 2021. BLSTM-based adaptive finite-time output-constrained control for a class of AUSs with dynamic disturbances and actuator faults. Mathematical Problems in Engineering, Art no. 2221495.

Huynh, T., Lin, C., Le, T., Cho, H., Pham, T.T., Le, N., Chao, F., 2020. A new self-organizing fuzzy cerebellar model articulation controller for uncertain nonlinear systems using overlapped Gaussian membership functions. IEEE Transactions on Industrial Electronics 67(11), 9671–9682.

Jiang, J., Astolfi, A., 2021. Stabilization of a class of underactuated nonlinear systems via underactuated back-stepping. IEEE Transactions on Automatic Control 66(11), 5429–5435.

Le, T.L., Ngo, V.B., 2022. The synchronization of hyperchaotic systems using a novel interval type-2 fuzzy neural network controller. IEEE Access 10, 105966–105982.

Lin, C.M., Nguyen, H.B., Huynh, T.T., 2021. A new self-organizing double function-link brain emotional learning controller for MIMO nonlinear systems using sliding surface. IEEE Access 9, 73826–73842.

Lin, C.T., Lee, C.S.G., 1996. Neural Fuzzy Systems- a neural-fuzzy synergism to intelligent systems. Englewood Cliffs, New Jersey, Prentice-Hall.

Slotine, J.J.E., Li, W.P., 1991. Applied nonlinear control. Prentice-Hall, Englewood Cliffs.

Sui, S., Chen, C.L.P., Tong, S., Feng, S., 2020. Finite-time adaptive quantized control of stochastic nonlinear systems with input quantization: a broad learning system based identification method. IEEE Transactions on Industrial Electronics 67(10), 8555–8565.

Tian, W., Zhao, F., Min, C., Feng, X., Liu, R., Mei, X., Chen, G., 2022. Broad learning system based on binary grey wolf optimization for surface roughness prediction in slot milling. IEEE Transactions on Instrumentation and Measurement 71, 1–10, Art no. 2502310.

Tsai, C.C., Chan, C.C., Li, Y.C., Tai, F.C., 2020. Intelligent adaptive PID control using fuzzy broad learning system: an application to tool-grinding servo control systems. International Journal of Fuzzy Systems 22, 2149–2162.

Wai, R.J., Lin, Y.F., Chuang, K.L., 2014. Total sliding-mode-based particle swarm optimization control for linear induction motor. Journal of the Franklin Institute 351(5), 2755–2780.

Wang, B., Zhao, Y., Chen, C.L.P., 2021. Hybrid transfer learning and broad learning system for wearing mask detection in the COVID-19 Era. IEEE Transactions on Instrumentation and Measurement 70, 1–12, Art no. 5009612.

Wang, C.H., Lin, T.C., Lee, T.T., Liu, H.L., 2002. Adaptive hybrid intelligent control for uncertain nonlinear dynamical systems. IEEE Transactions on Systems, Man, and Cybernetics 32(5), 583–597.

Wang, L.X., 1994. Adaptive Fuzzy Systems and Control: Design and Stability Analysis. Englewood Cliffs, NJ: Prentice-Hall.

Wang, X., Huang, T., Zhu, K., Zhao, X., 2022. LSTM-based broad learning system for remaining useful life prediction. Mathematics 10(12), Art no. 2066.

Xu, S., Liu, J., Yang, C., Wu, X., Xu, T., 2022. A learning-based stable servo control strategy using broad learning system applied for microrobotic control. IEEE Transactions on Cybernetics 52(12), 13727–13737.

Yi, J., Huang, J., Zhou, W., Chen, G., Zhao, M., 2022. Intergroup cascade broad learning system with optimized parameters for chaotic time series prediction. IEEE Transactions on Artificial Intelligence 3(5), 709–721.

Yuan, L., Li, T., Tong, S., Xiao, Y., Shan, Q., 2022. Broad learning system approximation-based adaptive optimal control for unknown discrete-time nonlinear systems. IEEE Transactions on Systems, Man, and Cybernetics: Systems 52(8), 5028–5038.

Zhang, J., Chao, F., Zeng, H., Lin, C.M., Yang, L., 2022. A recurrent wavelet-based brain emotional learning network controller for nonlinear systems. Soft Computing 26, 3013–3028.

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 Transactions on Mechatronics 25(4), 1904–1911.

Zhang, Q.Q., Wai, R.J., 2022. Design of adaptive distributed secondary control using double-hidden-layer recurrent-neural-network-inherited total-sliding-mode scheme for islanded micro-grid. IEEE Access 10, 5990–6009.

Zhao, J., Lin, C.M., 2019. Wavelet-TSK-type fuzzy cerebellar model neural network for uncertain nonlinear systems. IEEE Transactions on Fuzzy Systems 27(3), 549–558.

Table 1: Parameter symbol definition of reaction wheel pendulum

ϕ	pendulum angle
τ	motor torque acting on the reaction wheel
m_p, m_r	pendulum mass, reaction wheel mass
l_p	pendulum length
l_{cp}	distance to the pendulum center of mass
g	gravity acceleration
J_p	pendulum moment of inertia
J_r	reaction wheel moment of inertia

Table 2: Characteristic comparison

	FNC	FBNC	HBRNC with low learning rates	HBRNC with high learning rates	HBRNC with ALR
robustness ability	nice	great	nice	excellent	excellent
stability proof	yes	yes	yes	yes	yes
learning ability	yes	yes	yes	yes	yes
learning speed	slow	middle	slow	fast	fast
implementation complex	middle	hard	hard	hard	hard
control chattering	no	no	no	yes	no

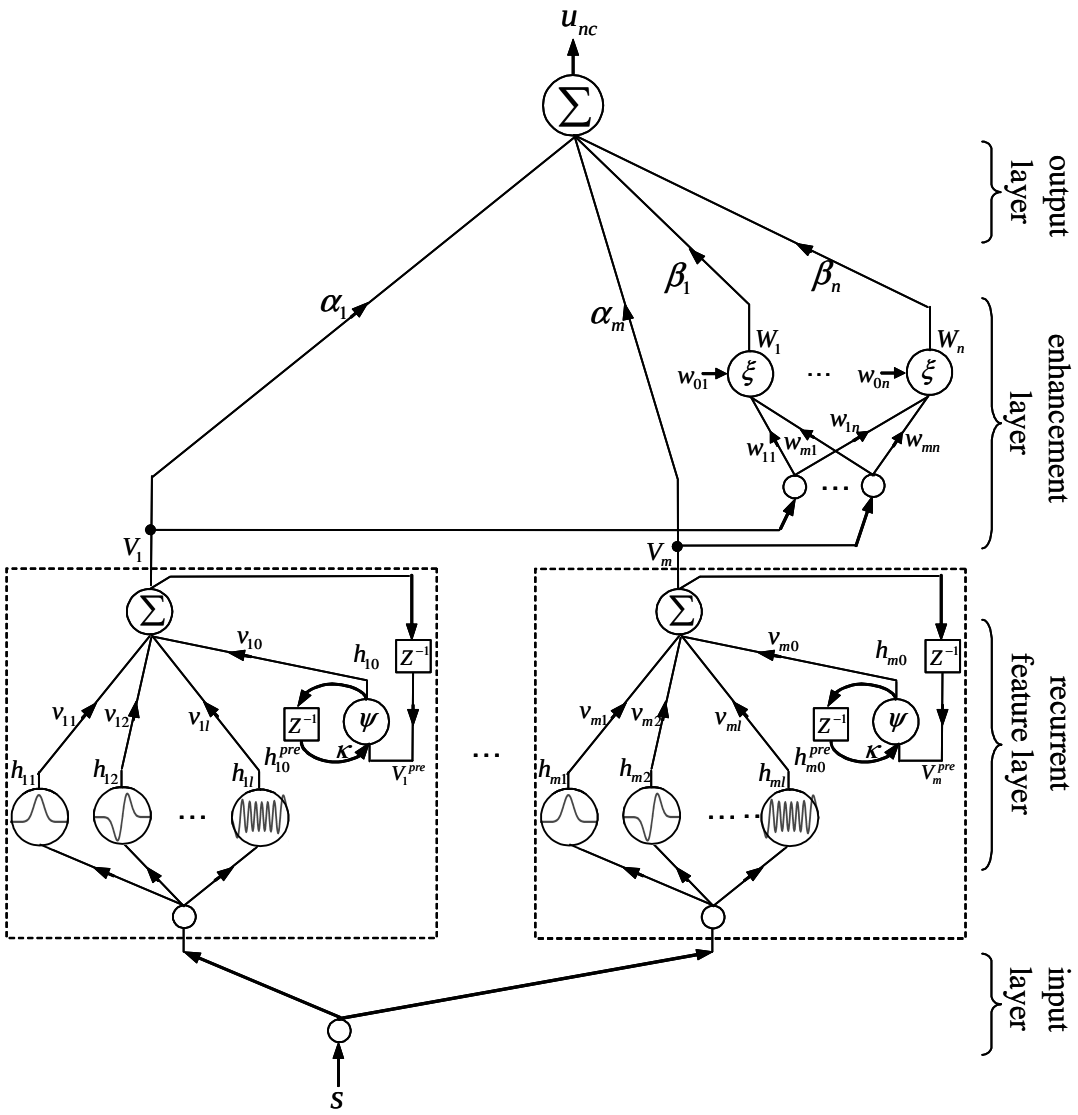

Figure 1: Network Structure of HBRNN

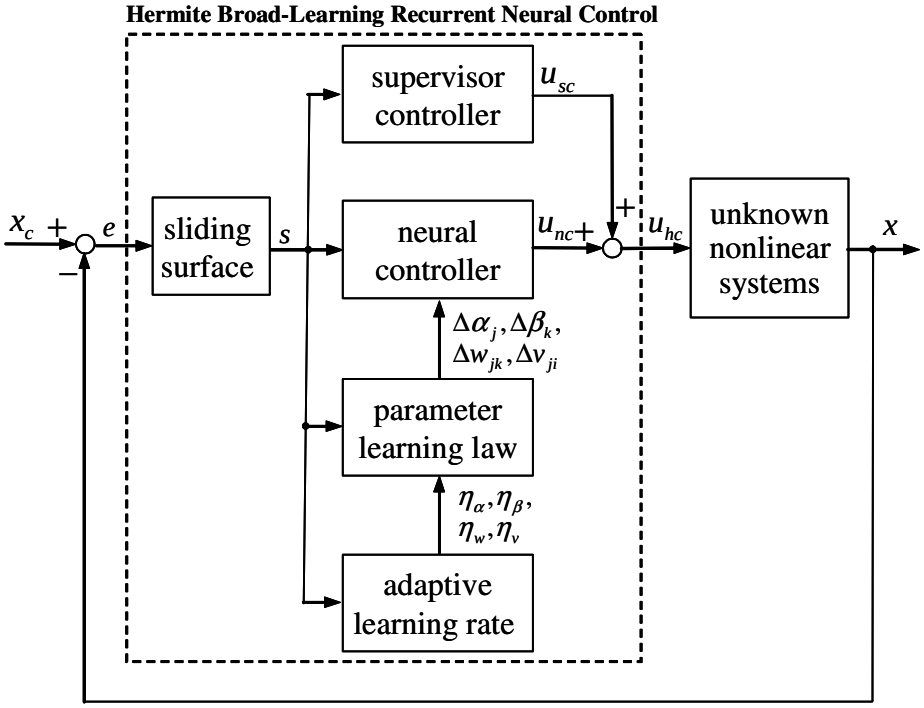

Figure 2: Block Diagram of the HBRNC system with ALR

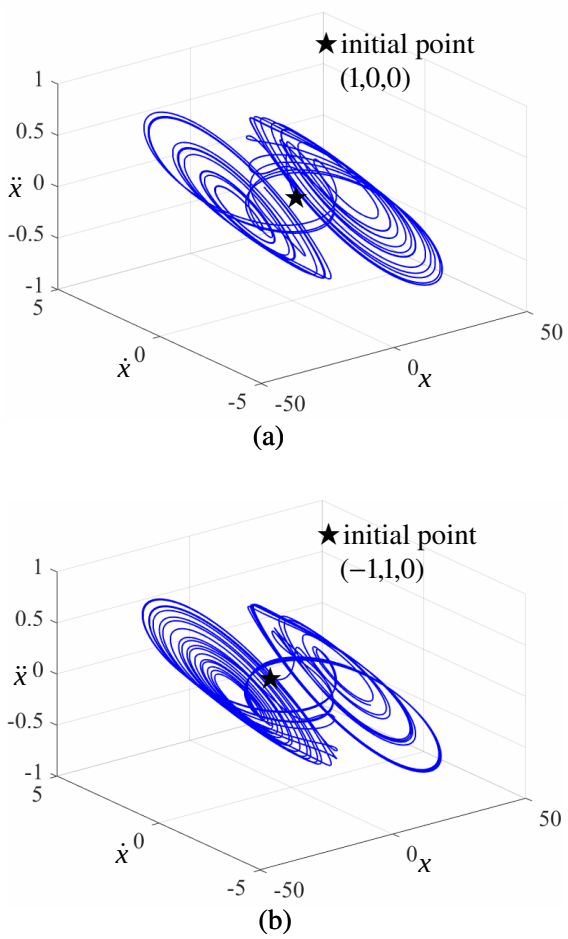

Figure 3: Behavior of Uncontrolled Chaotic Circuit
(a) initial point (1,0,0); (b) initial point (−1,1,0)

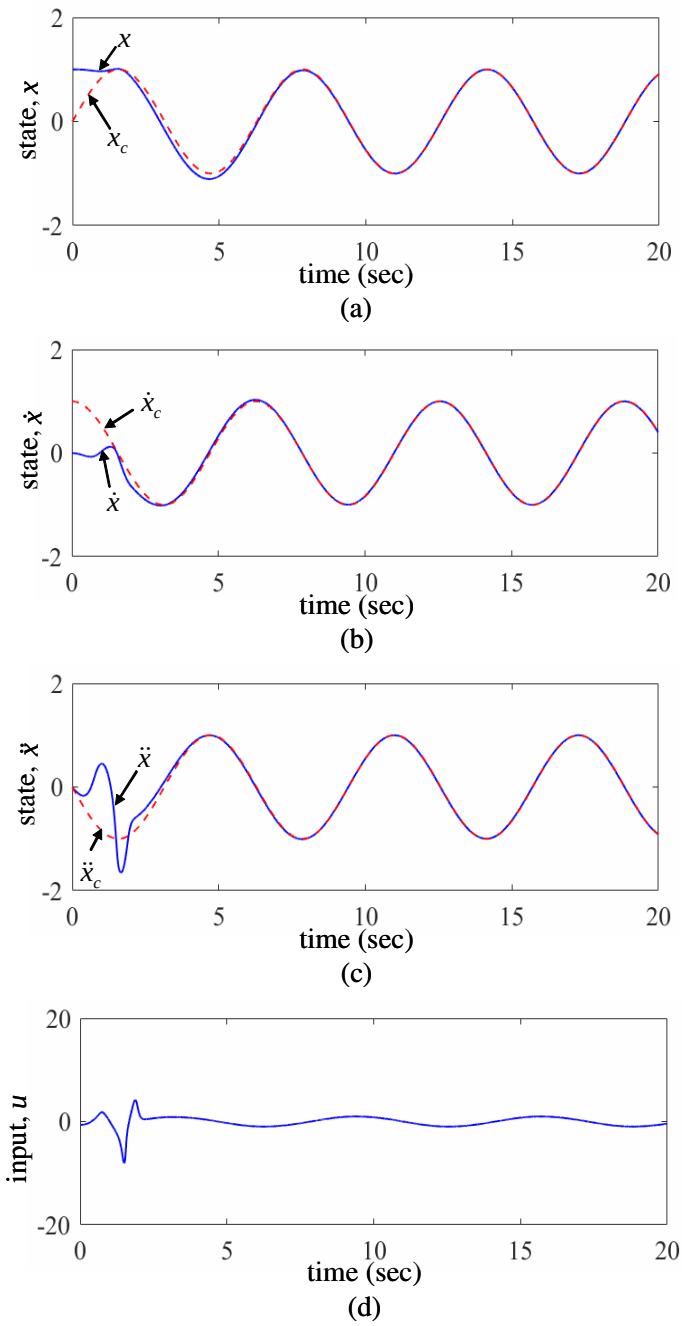

Figure 4: (Cont.)

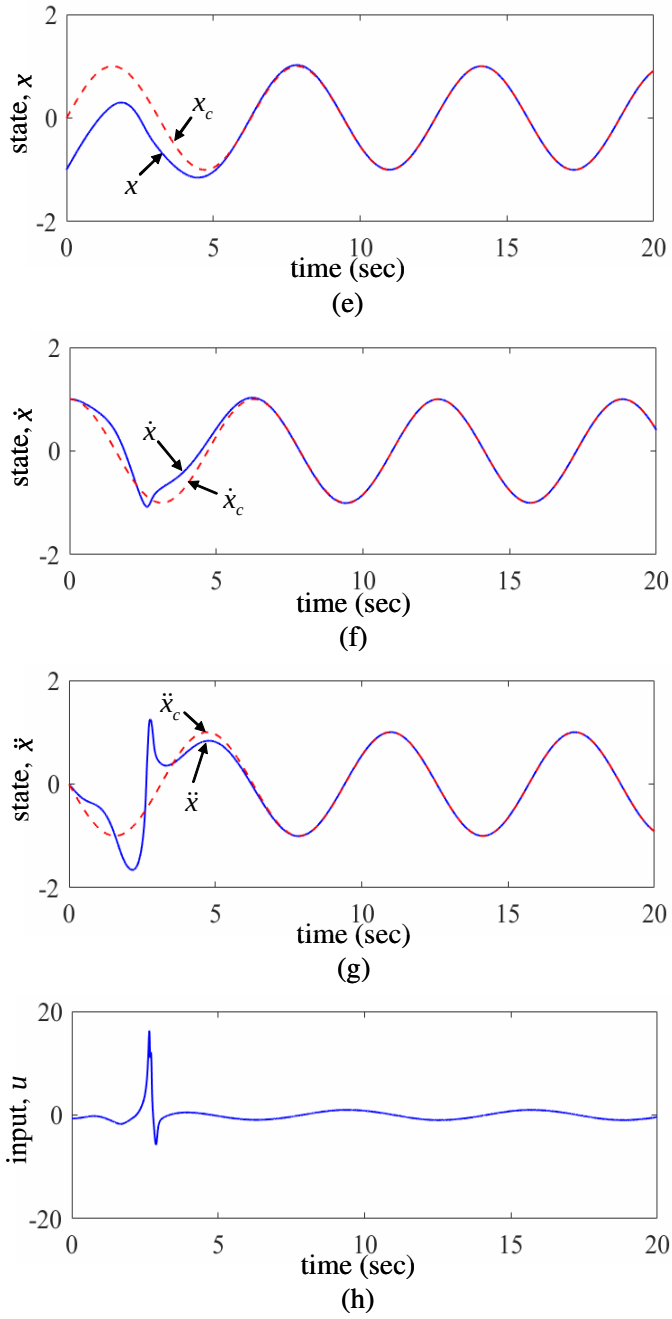

Figure 4: Simulation Results of the HBRNC System with Low Learning Rates (a)–(d) initial point (1,0,0); (e)–(h) initial point (−1,1,0)

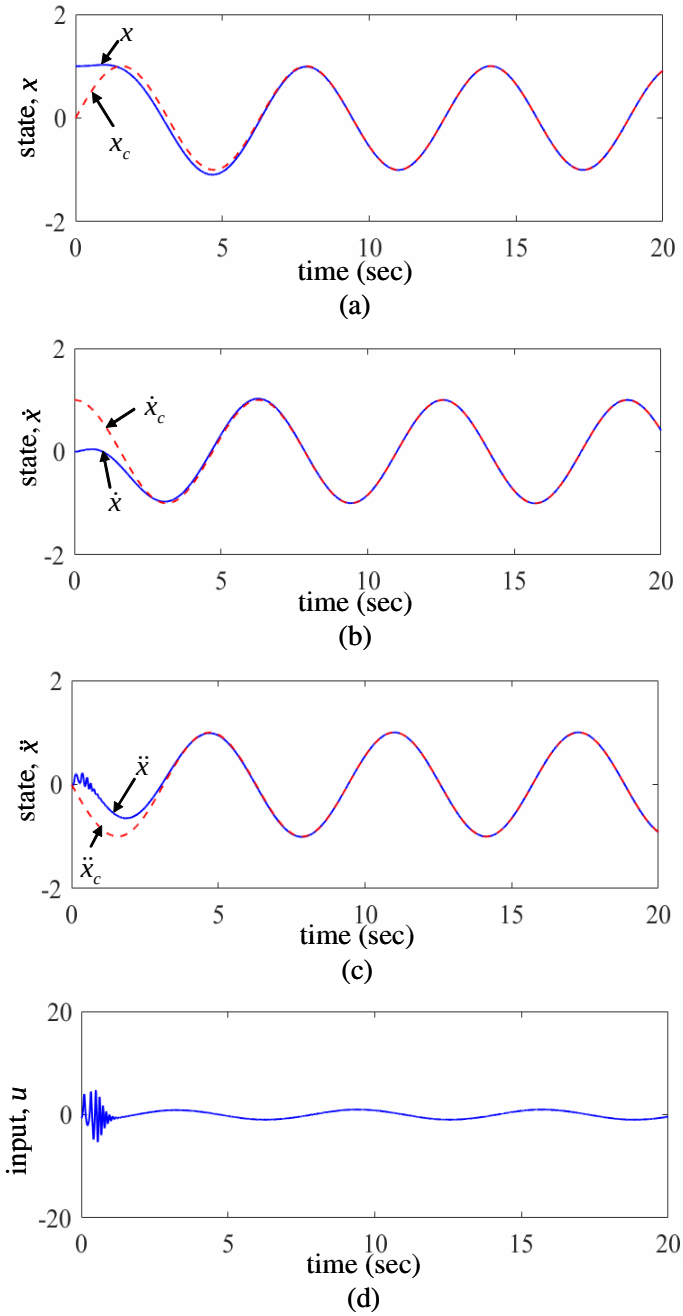

Figure 5: (Cont.)

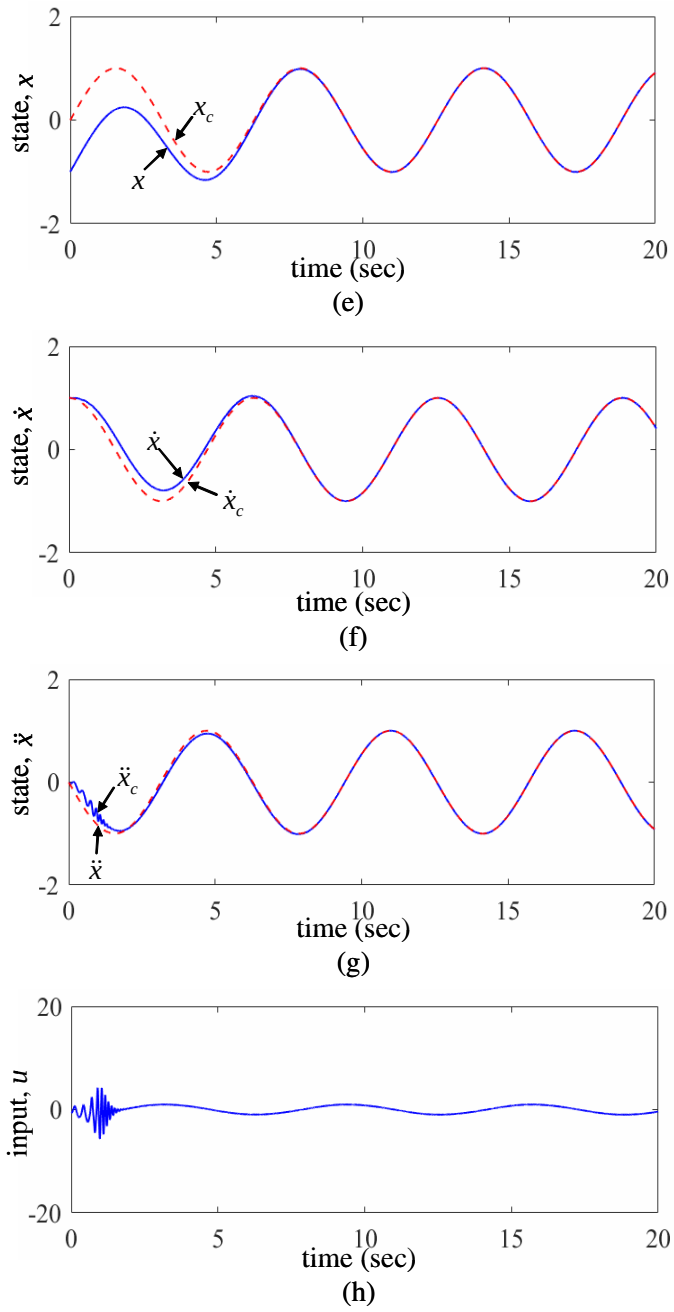

Figure 5: Simulation Results of the HBRNC System with High Learning Rates (a)–(d) initial point (1,0,0); (e)–(h) initial point (−1,1,0)

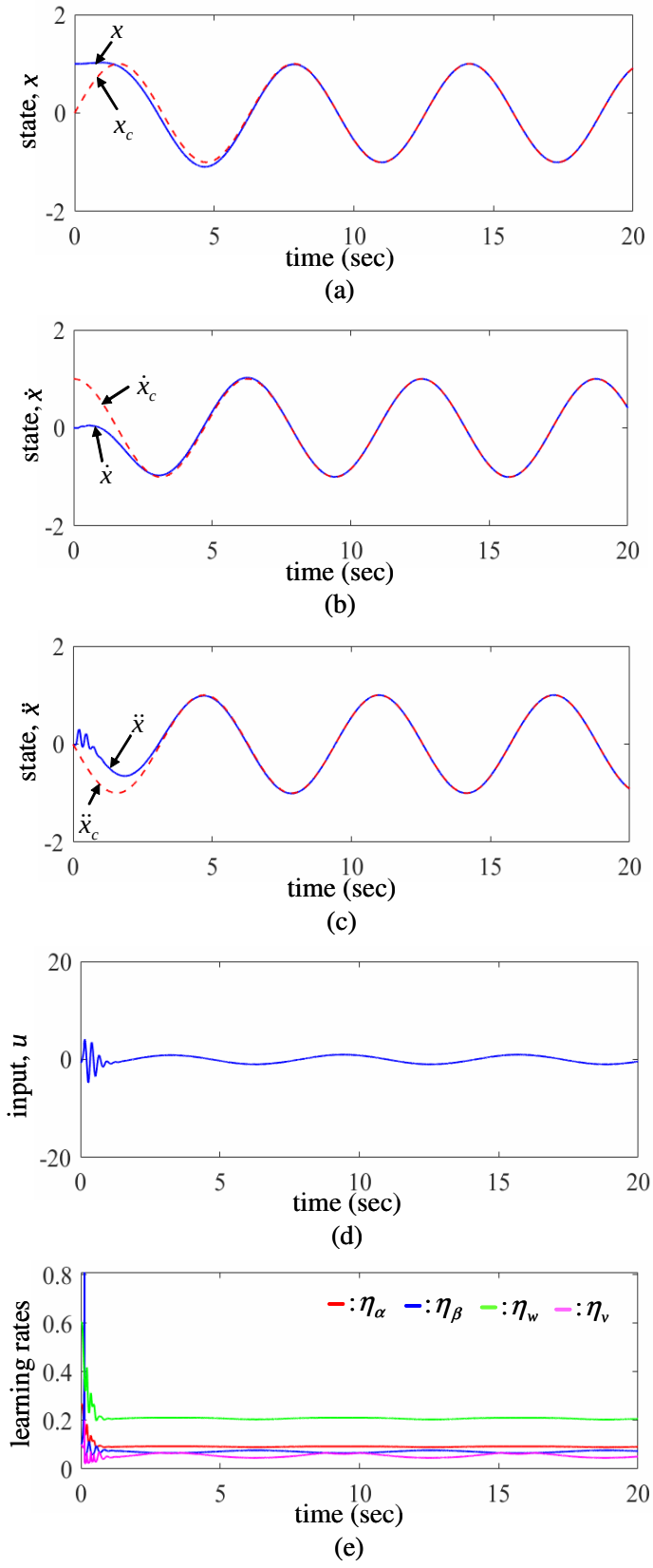

Figure 6: (Cont.)

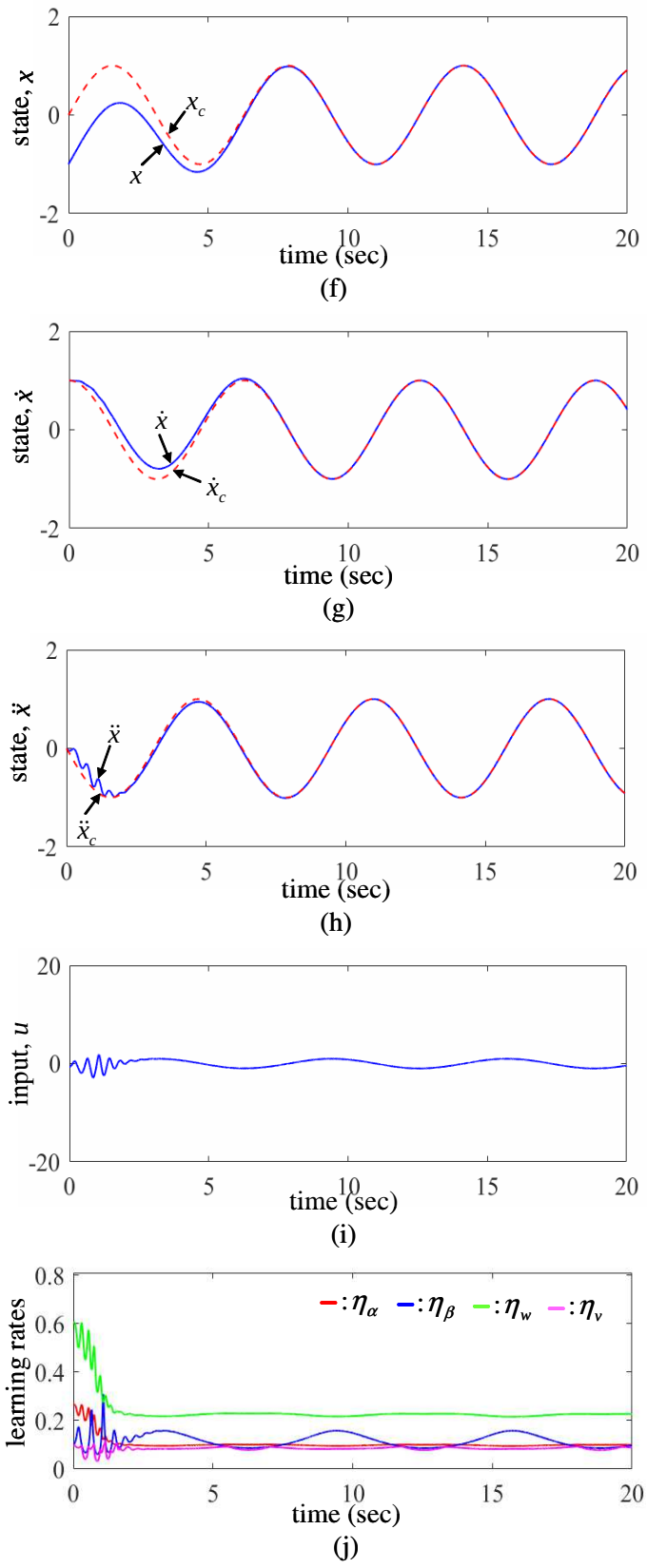

Figure 6: Simulation Results of the HBRNC System with ALR
(a)–(e) initial point $(1,0,0)$; (f)–(j) initial point $(-1,1,0)$

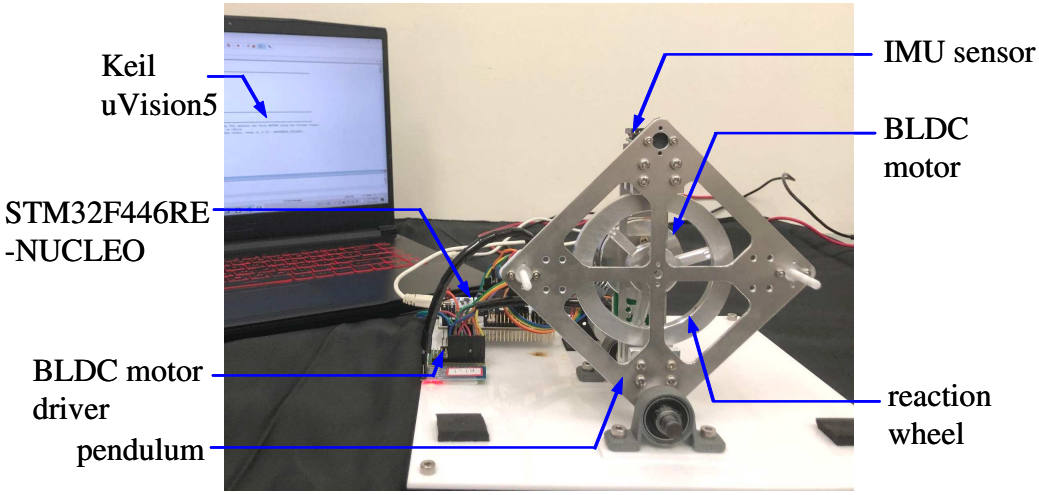

Figure 7: Microcontroller-Based Experimental Setup

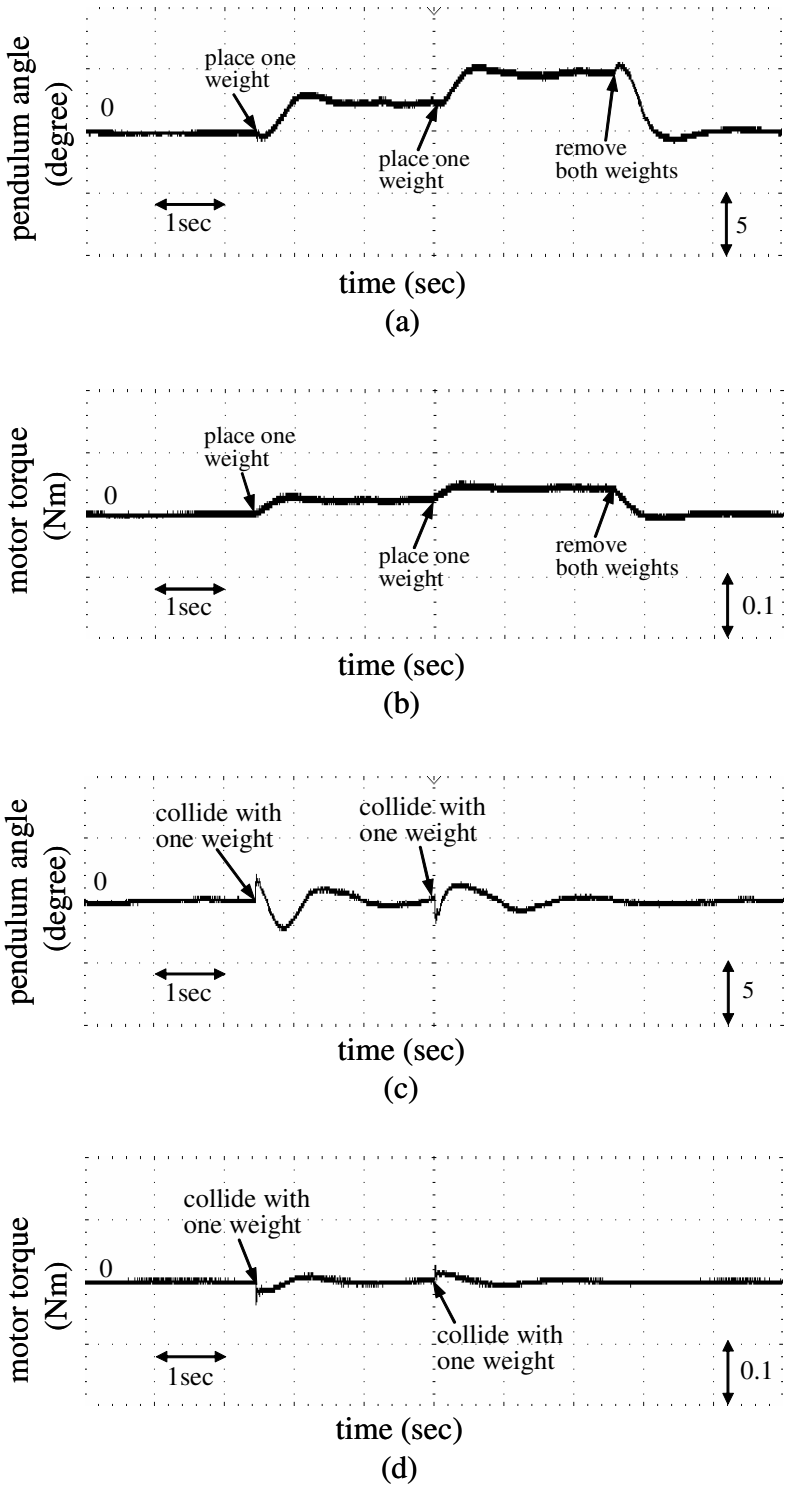

Figure 8: Experimental Results of the AFNC System
(a),(b) payload scenario; (c),(d) disturbance scenario

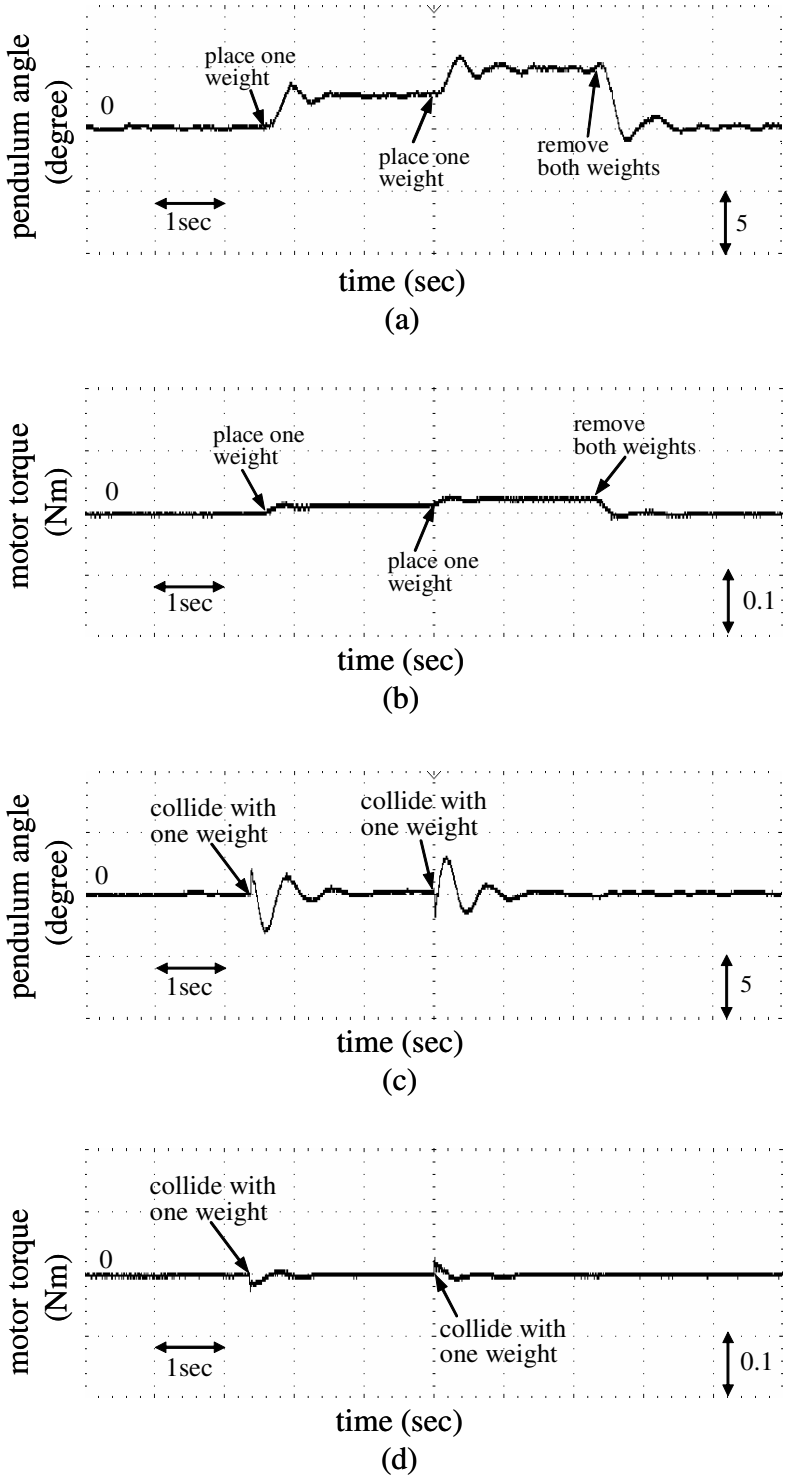

Figure 9: Experimental Results of the AFBNC System
(a),(b) payload scenario; (c),(d) disturbance scenario

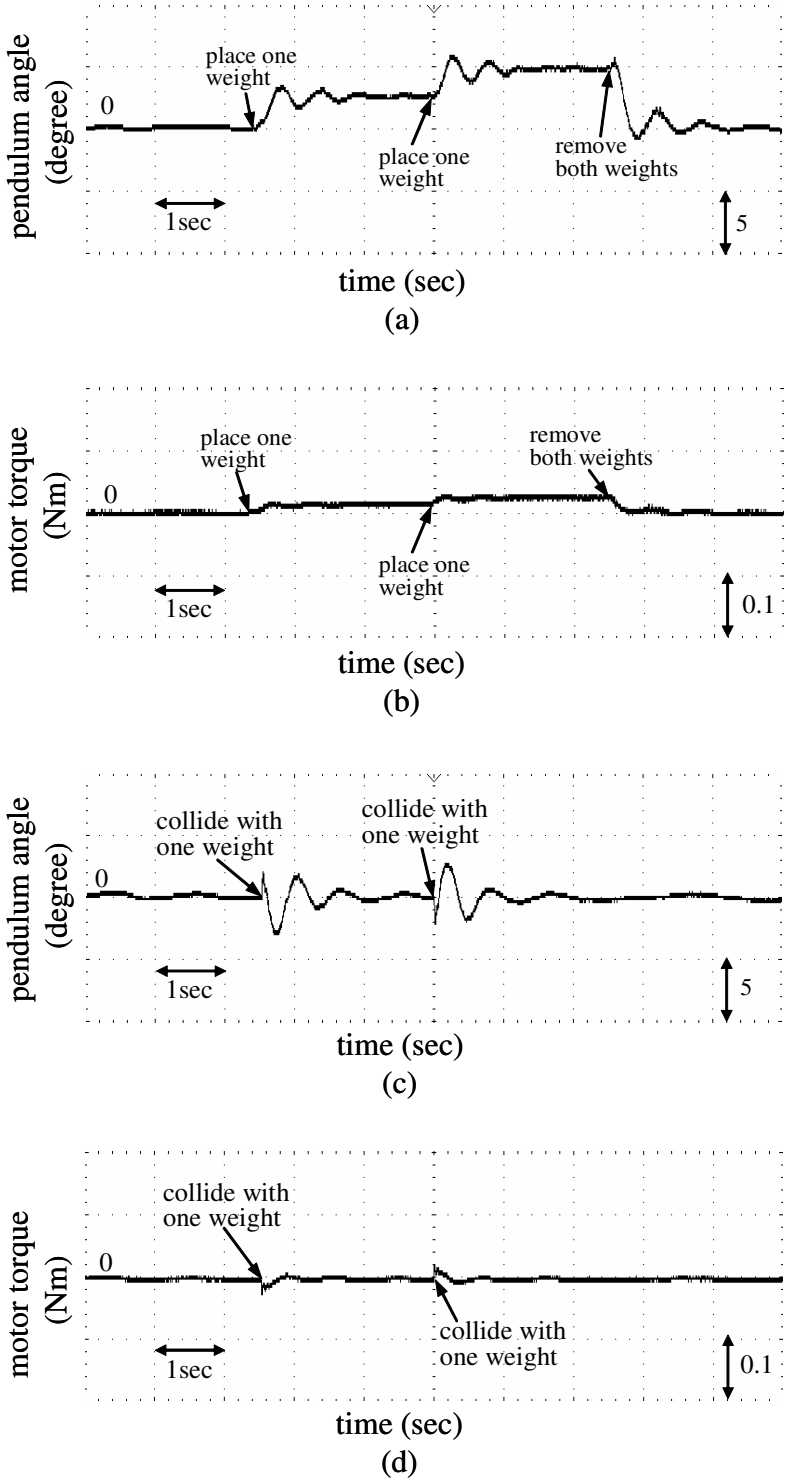

Figure 10: Experimental Results of the HBRNC System with Low Learning Rates
(a),(b) payload scenario; (c),(d) disturbance scenario

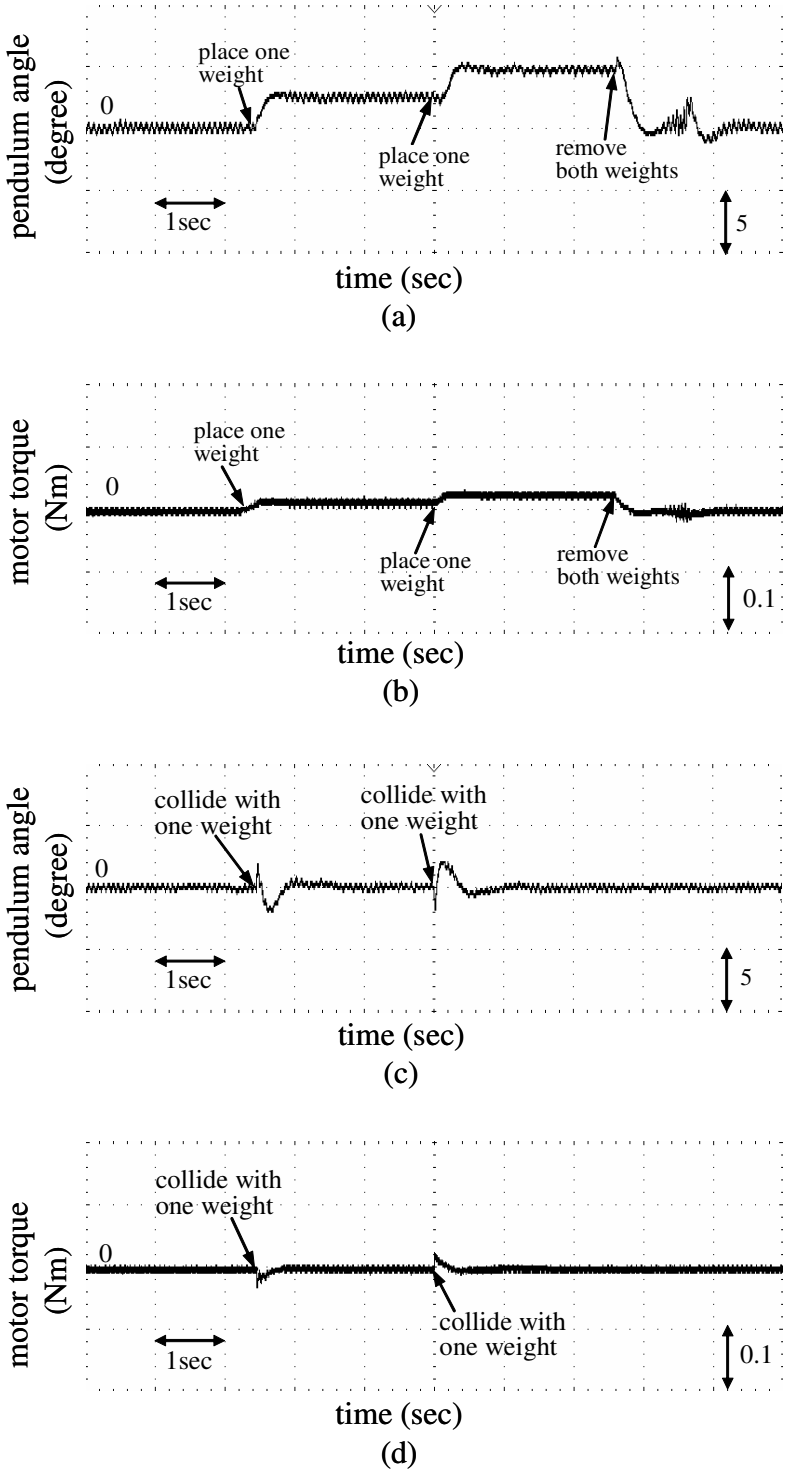

Figure 11: Experimental Results of the HBRNC System with High Learning Rates (a),(b) payload scenario; (c),(d) disturbance scenario

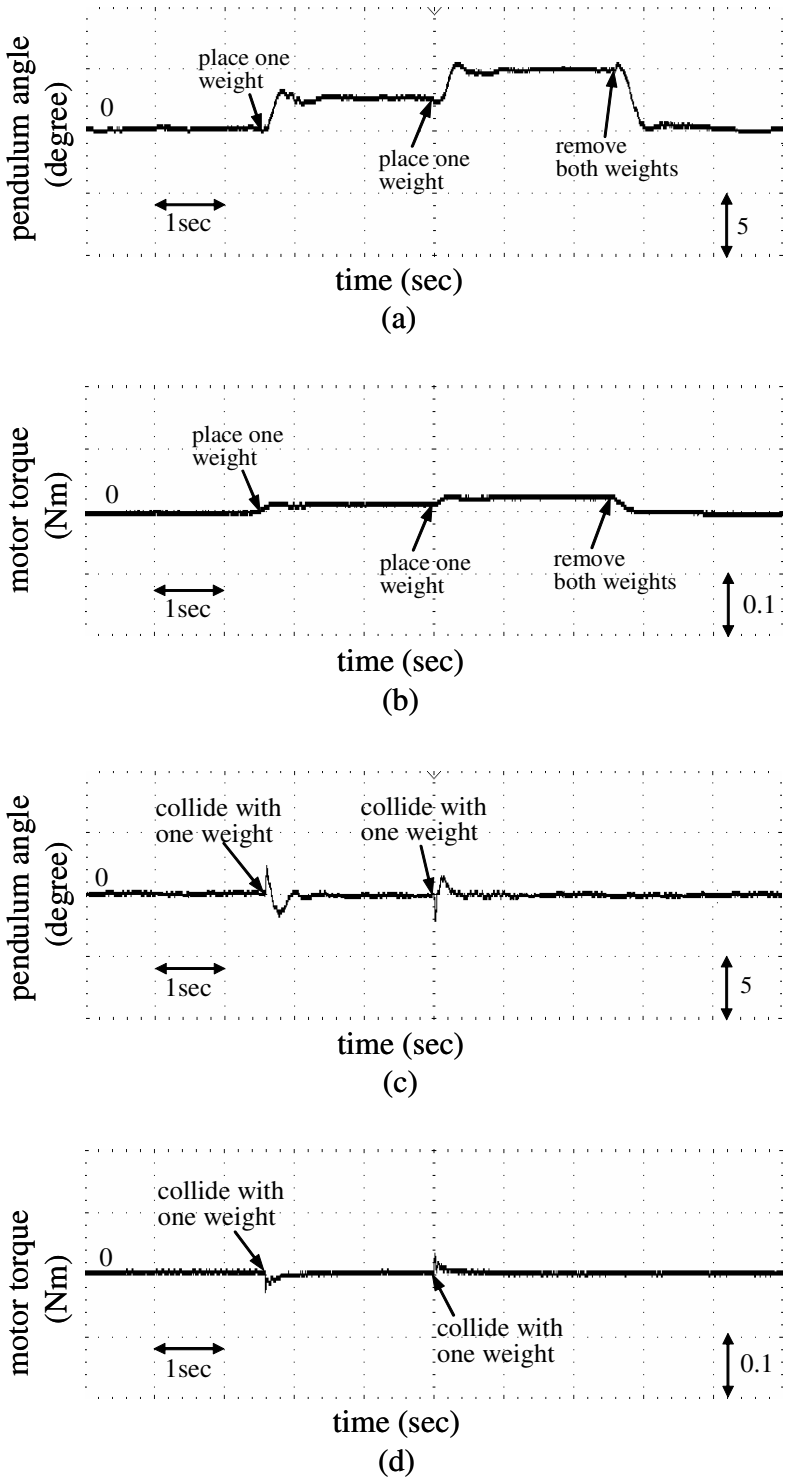

Figure 12: Experimental Results of the HBRNC System with ALR
(a),(b) payload scenario; (c),(d) disturbance scenario