Journal of Mathematical Modeling
Vol. 10, No. 3, 2022, pp. 529-547. Research Article

JMM

Stochastic gradient-based hyperbolic orthogonal neural networks for nonlinear dynamic systems identification

Ghasem Ahmadi*

Department of Mathematics, Payame Noor University, P.O. Box 19395-4697, Tehran, Iran
Email(s): g.ahmadi@pnu.ac.ir

Abstract. Orthogonal neural networks (ONNs) are some powerful types of the neural networks in the modeling of non-linearity. They are constructed by the usage of orthogonal functions sets. Piecewise continuous orthogonal functions (PCOFs) are some important classes of orthogonal functions. In this work, based on a set of hyperbolic PCOFs, we propose the hyperbolic ONNs to identify the nonlinear dynamic systems. We train the proposed neural models with the stochastic gradient descent learning algorithm. Then, we prove the stability of this algorithm. Simulation results show the efficiencies of proposed model.

Keywords: System identification, piecewise continuous orthogonal functions, hyperbolic orthogonal neural networks, stochastic gradient descent.
AMS Subject Classification 2010: 68T05, 93B30.

1 Introduction

Identification of nonlinear dynamic systems is an important and attractive field of control engineering. In this context, the models are provided using the measurable input and output data of underlying systems. To design the controllers for nonlinear dynamic systems, we need to know their models where achieving the reliable models using the first principles is very hard [26]. Nevertheless, the measurement of their input-output data is possible easily. By attention to the recent improvements in computational intelligence, there is an increasing interest to develop the new methodologies for system identification based on these approaches. Artificial neural networks (ANNs) are potent instruments in this context.

ANNs are some computational structures that include many parameters, and they are trained with some learning algorithms such that the neural structure can model the underlying system [17]. ANNs are universal approximators, and they have an inherent potential for parallel computations. They have been used for different purposes, such as classification, prediction, and system identification.

*Corresponding author.
Received: 17 January 2022 / Revised: 21 April 2022/ Accepted: 6 May 2022
DOI: 10.22124/JMM.2022.21572.1890

In recent decades, many efforts have been made to benefit the outstanding properties of neural networks in the identification and control of nonlinear systems. In 1989, Chen et al. [9] used neural networks for system identification. Narendra and Parthasarathy [25] utilized neural networks to identify the nonlinear systems. Abdollahi et al. [1] proposed some stable algorithms to identify the nonlinear systems with neural networks. Yu [33] designed the discrete-time recurrent neural networks with stable learning algorithms to identify the nonlinear systems. Ahmadi et al. [2–5] utilized rough-neural networks with stable Lyapunov-based and stochastic gradient-based learning algorithms to identify the different types of nonlinear systems. Sahoo and Chakraverty [30] used the functional link neural networks to solve the structural system identification problems. Forgione and Piga [14] employed the neural networks to identify the continuous-time nonlinear systems. Tavoosi et al. [31] published a review on the type-2 fuzzy neural identifiers.

Some important types of neural networks are the orthogonal neural networks (ONNs). They are constructed based on orthogonal functions (OFs) sets. OFs are some exiting and valuable sets in function spaces with an elegant and robust mathematical theory. They have a remarkable ability in function approximation. However, most of the OFs that have been used in designing neural networks are orthogonal polynomials.

Piecewise continuous OFs (PCOFs) are essential classes of OFs in the literature. Datta and Mohan [10] described the orthogonal functions and their applications in systems and control. Babolian and Salimi Shamloo [6] applied the piecewise constant orthogonal functions for the numerical solution of Volterra integral and integro-differential equations.

The set of block pulse functions (BPFs) is a well-known set of OFs [13]. Using BPFs, some other sets of OFs have been proposed, such as triangular functions [11], hybrid functions [13] and sinusoidal basis functions [32]. Some of the OFs have been used for the analysis of nonlinear systems [12, 16]. Besides, some of them have been used for system identification [13]. Heydari and Razzaghi [15] proposed the piecewise Chebyshev cardinal functions to solve the constrained fractional optimal control problems. Recently, based on BPFs, a set of sinusoidal PCOFs has been introduced [32].

Orthogonal neural networks (ONNs) are some consequential types of neural networks in the modeling of non-linearity. They are constructed by the usage of orthogonal functions sets [35]. They have simple structures, and due to the existence of a robust mathematical theory for the approximation capabilities of OFs, their performances in system identification are outstanding. In the last years, Yang and Tseng [35] used the Legendre ONNs for function approximation. Lee and Jeng [22] used the ONN with Chebyshev polynomial basis function for function approximations. Purwar et al. [28] used the ONNs with the Chebyshev polynomials as the orthogonal basis for system identification. Kumar et al. [21] have been compared the different types of ONNs for system identification. Vukovic et al. [34] gave a comprehensive experimental evaluation of orthogonal polynomials in the ONNs.

Training the neural networks is a fundamental issue in their applications. Some of the well-known learning algorithms are the standard gradient-based algorithms. Sometimes, these algorithms do not converge to the global minimum of the cost function. Recently, to avoid the deficiencies of these algorithms, some stable learning algorithms have been proposed by the authors. Yu [33] developed some stable learning algorithms for discrete-time recurrent neural networks to identify nonlinear systems. Man et al. [23] proposed a Lyapunov-based learning algorithm for multilayer perceptron. Janakiraman et al. [20] suggested a Lyapunov-based learning algorithm for extreme learning machines. Ahmadi et al. [2, 3, 5] proposed some Lyapunove-based learning algorithm for rough-neural identifiers.

Besides, some works have been published about the usage of stochastic gradient descent (SGD)

learning algorithms for neural networks. SGD is a robust algorithm with frequent application in machine learning. In SGD, the sampled data are arbitrarily presented to the model one by one, and the parameters are updated using the gradient of the cost function for each sample. For the first time, this algorithm is introduced by Robbins and Monro [29]. For large datasets, SGD is faster and more reliable than the standard gradient descent, and due to its stochastic behavior, usually, it achieves the global minimum of the cost function [7, 8]. Recently, Janakiraman et al. [19] utilized the SGD to train the extreme learning machines in the online training of advanced combustion engines. Netrapalli reviewed the variants of SGD in machine learning [27].

Recently, Ahmadi and Teshnehlab [4] utilized the rough-neural networks (R-NNs) for to identify the cement rotary kiln (CRK). R-NNs are some extensions of multilayer perceptron designed based on rough set theory to deal with the uncertainties in neural networks. In [4], the R-NNs are trained by the SGD learning algorithm. In the present work, based on a complementary pair of PCOF sets, a new orthogonal neural network is proposed for the identification of nonlinear systems. This neural network is inherently different from R-NNs in [4]. To train the proposed model, the SGD algorithm is employed due to its specific properties, as mentioned above.

PCOFs have good properties in function approximations. In the approximation of nonlinear function $f(t)$ with OFs, according to the related formula, we need to compute the coefficients from the integration of f, wherein most of the applications, it is unknown [35]. Hence, it is necessary to use the models that their formulations can not depend on unknown functions. As mentioned above, ANNs have a significant role in this context. The capabilities of ANNs in nonlinear system identification are apparent. This work tries to introduce a new ONN based on some families of PCOFs to identify the nonlinear systems. On the basis of hyperbolic functions, we introduce a new set of PCOFs and utilize it to design the hyperbolic ONNs (HONNs). We apply the proposed ONNs to identify the nonlinear dynamic systems where their parameters are adjusted with a stable SGD algorithm.

We can summarize the innovations of this work as follow:

- On the basis of hyperbolic functions, we introduce a new set of PCOFs.
- Using the introduced set of PCOFs, we propose the HONNs, and we use them to identify the nonlinear dynamic systems.
- We train the proposed neural structures using a SGD learning algorithm, and prove its stability.

We continue the paper as follows. In Section 2, we introduce the hyperbolic PCOFs. We propose the HONNs in the Section 3. Section 4 describes the SGD learning algorithm and its stability analysis proposed for HONNs. System identification of nonlinear systems with the proposed method and their simulations are presented in Section 5. We draw the conclusion in Section 6.

2 Hyperbolic Piecewise Continuous Orthogonal Functions

In the literature, some sets of PCOFs have been introduced such as block pulse functions, triangular orthogonal functions [11], and sinusoidal PCOFs [32]. Here, we introduce the hyperbolic PCOFs (HPCOFs) and consider some of their properties. We define a set of HPCOFs with m members on the interval $[0,T)$ as follows:

$$H1_i(t) = \begin{cases} \cosh\left(\frac{m}{2T}(t - i\frac{T}{m})\right), & \frac{iT}{m} \leq t < \frac{(i+1)T}{m}, \\ 0, & O.W. \end{cases} \tag{1}$$

and

$$H2_i(t) = \begin{cases} \sinh\left(\frac{m}{2T}(t - i\frac{T}{m})\right), & \frac{iT}{m} \le t < \frac{(i+1)T}{m}, \\ 0, & O.W., \end{cases} \tag{2}$$

for $i = 0,1,2,\ldots,m-1$. We introduce the following vectors of HPOFs for each $t \in [0,T)$:

$$\begin{aligned} \mathbf{H1}(t) &= [H1_0(t), H1_1(t), \ldots, H1_{m-1}(t)]^T, \\ \mathbf{H2}(t) &= [H2_0(t), H2_1(t), \ldots, H2_{m-1}(t)]^T, \\ \mathbf{H}(t) &= [\,\mathbf{H1}(t),\, \mathbf{H2}(t)]^T. \end{aligned}$$

For the orthogonality of the components of $\mathbf{H1}(t)$, it is necessary that [11]:

$$\int_0^T H1_p(t)H1_q(t)dt = \begin{cases} \text{constant}, & p = q, \\ 0, & p \neq q. \end{cases}$$

Due to the mutually disjointness of these functions, we have

$$H1_p(t)H1_q(t) = 0 \quad \text{for } p \neq q.$$

Therefore,

$$\int_0^T H1_p(t)H1_q(t)dt = 0, \quad \text{for } p \neq q.$$

For $p = q$, we have

$$\begin{aligned} \int_0^T [H1_p(t)]^2 dt &= \int_{ph}^{(p+1)h} [H1_p(t)]^2 dt = \int_{ph}^{(p+1)h} \left(\cosh\left(\frac{1}{2h}(t - ph)\right)\right)^2 dt \\ &= \frac{1}{4}h\left(e - e^{-1} + 2\right) \text{ (constant)}, \end{aligned}$$

where $h = T/m$. Similar to this discussion, the orthogonality of the components of $\mathbf{H2}(t)$ can be proved. Also, we have

$$H1_p(t)H2_q(t) = 0 \quad \text{for } p \neq q.$$

Therefore,

$$\int_0^T H1_p(t)H2_q(t)dt = 0 \quad \text{for } p \neq q.$$

For $p = q$, we have

$$\begin{aligned} \int_0^T (H1_p(t)H2_p(t))\, dt &= \int_{ph}^{(p+1)h} (H1_p(t)H2_p(t))\, dt \\ &= \int_{ph}^{(p+1)h} \left(\cosh\left(\frac{1}{2h}(t - ph)\right)\right)\left(\sinh\left(\frac{1}{2h}(t - ph)\right)\right) dt \\ &= \frac{1}{4}h\left(e + e^{-1} - 2\right) \text{ (constant)}. \end{aligned}$$

Therefore, the orthogonality of HPCOFs is proved.

Let $f(t)$ be a square Lebesgue integrable function. Then, f can be expanded into a HPCOF series:

$$f(t) = \sum_{i=0}^{m-1} c_i H1_i(t) + \sum_{i=0}^{m-1} d_i H2_i(t) = C^T \mathbf{H1}(t) + D^T \mathbf{H2}(t)], \tag{3}$$

where $C = [c_1, c_2, \ldots, c_{m-1}]$, $D = [d_1, d_2, \ldots, d_{m-1}]$, and the coefficients c_i and d_i, $i = 0, 1, \ldots, m-1$, are

$$c_i = \frac{1}{h} \int_{ih}^{(i+1)h} f(t) H1_i(t) dt, \quad d_i = \frac{1}{h} \int_{ih}^{(i+1)h} f(t) H2_i(t) dt, \tag{4}$$

where $h = T/m$.

In (3), the function f is supposed to be single-variable. For the approximation of multi-variable functions, by integrating the single-variable orthogonal functions (for example, multiplication of them), we can generate a set of multi-variable orthogonal functions [35].

Remark 1. *In (4), to compute the parameters c_i and d_i, it is necessary to know the function f where in the system identification, the function f is unknown. In this work, the proposed neural networks can model the unknown nonlinear dynamic systems where their parameters are trained with a stable learning algorithm.*

3 Hyperbolic Orthogonal Neural Network (HONN)

In this section, the structure of HONN is introduced. Let $X = [x_1, x_2, \ldots, x_n]^T$, and $Y = [y_1, y_2, \ldots, y_q]^T$ are the input and output vector of HONN, respectively. Further, let also $H1_i(t)$ and $H2_i(t)$ for $i = 0, 1, 2, \ldots, m-1$ be the i-th component of $\mathbf{H1}(t)$, and $\mathbf{H2}(t)$, respectively. In addition, suppose that

$$\begin{aligned} H1_i(X) &= [H1_i(x_1), H1_i(x_2), \ldots, H1_i(x_n)]^T, \quad i = 0, 1, 2, \ldots, m-1, \\ H2_i(X) &= [H2_i(x_1), H2_i(x_2), \ldots, H2_i(x_n)]^T, \quad i = 0, 1, 2, \ldots, m-1, \end{aligned}$$

$W1_i$, $i = 0, 1, 2, \ldots, m-1$ is a $q \times n$ matrix of weights between the nodes of $H1_i(X)$ and outputs, and $W2_i$, $i = 0, 1, 2, \ldots, m-1$ is a $q \times n$ matrix of weights between the nodes of $H2_i(X)$ and outputs. Then, we have

$$Y = \sum_{i=0}^{m-1} W1_i H1_i(X) + \sum_{i=0}^{m-1} W2_i H2_i(X) = W1\mathbf{H1}(X) + W2\mathbf{H2}(X),$$

where

$$\begin{aligned} W1 &= [W1_0, W1_1, \ldots, W1_{m-1}], \\ W2 &= [W2_0, W2_1, \ldots, W2_{m-1}], \\ \mathbf{H1}(X) &= \left[H1_0^T(X), H1_1^T(X), \ldots, H1_{m-1}^T(X)\right]^T \\ &= [H1_0(x_1), H1_0(x_2), \ldots, H1_0(x_n), H1_1(x_1), H1_1(x_2), \ldots, H1_1(x_n), \\ &\quad \ldots, H1_{m-1}(x_1), H1_{m-1}(x_2), \ldots, H1_{m-1}(x_n)]^T, \\ \mathbf{H2}(X) &= \left[H2_0^T(X), H2_1^T(X), \ldots, H2_{m-1}^T(X)\right]^T \\ &= [H2_0(x_1), H2_0(x_2), \ldots, H2_0(x_n), H2_1(x_1), H2_1(x_2), \ldots, H2_1(x_n), \\ &\quad \ldots, H2_{m-1}(x_1), H2_{m-1}(x_2), \ldots, H2_{m-1}(x_n)]^T. \end{aligned}$$

Figure 1 shows the structure of HONN.

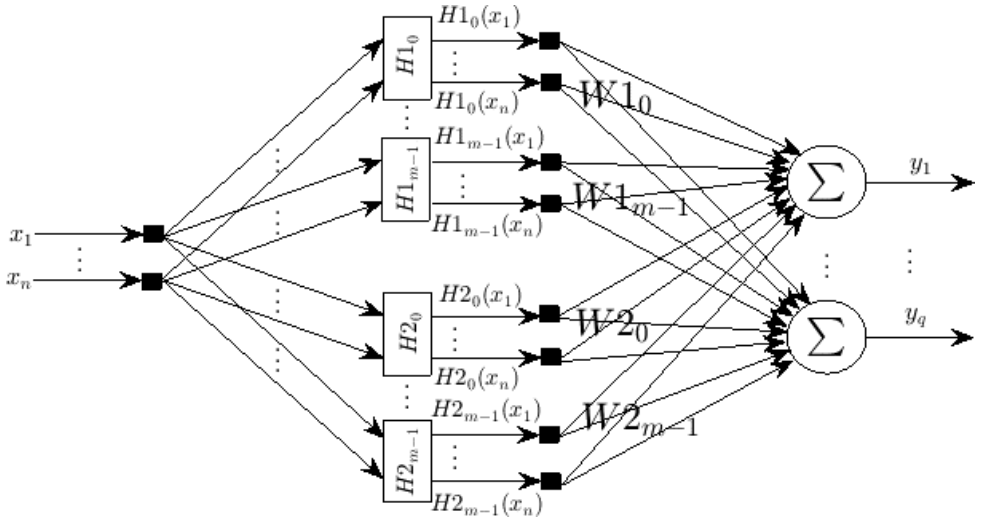

Figure 1: The structure of hyperbolic orthogonal neural network.

4 Stochastic Gradient Descent Learning Algorithm

In this Section, we propose an SGD learning algorithm to train the HONN. SGD is a powerful learning algorithm with frequently applications in machine learning. For large datasets, SGD is faster and more reliable than the standard gradient descent, and due to its stochastic behavior, usually, it achieves the global minimum of the cost function [7, 8]. Recently, SGD has been used to train the extreme learning machines and the R-NNs [4, 19].

Consider the input-output dataset $\{(X(k),Y(k)),\ i=1,2,\ldots,N\}$, where they are presented to HONN one after the other and randomly. Let $\widehat{Y}(k)$ be the model response to the input $X(k)$ and $E(k)=Y(k)-\widehat{Y}(k)$ be the model error. Here, the cost function is defined as follows:

$$J(W1,W2) \quad = \quad \min \sum_{k=1}^{N} J(W1(k),W2(k)) = \min \frac{1}{2} \sum_{k=1}^{N} \|E(k)\|^2 \tag{5}$$

Therefore, we have

$$\begin{aligned} J(k) &= J(W1(k),W2(k)) = \frac{1}{2}\|E(k)\|^2 = \frac{1}{2}\|Y(k)-\widehat{Y}(k)\|^2 \\ &= \frac{1}{2}\|Y(k)-W1(k)\mathbf{H1}(k)-W2(k)\mathbf{H2}(k)\|^2 \\ &= \frac{1}{2}\left(Y(k)-W1(k)\mathbf{H1}(k)-W2(k)\mathbf{H2}(k)\right)^T\left(Y(k)-W1(k)\mathbf{H1}(k)-W2(k)\mathbf{H2}(k)\right) \\ &= \frac{1}{2}Y(k)^TY(k)-Y(k)^TW1(k)\mathbf{H1}(k)-Y(k)^TW2(k)\mathbf{H2}(k)+\frac{1}{2}\mathbf{H1}(k)W1(k)^TW1(k)\mathbf{H1}(k) \\ &\quad +\mathbf{H1}(k)W1(k)^TW2(k)\mathbf{H2}(k)+\frac{1}{2}\mathbf{H2}(k)W2(k)^TW2(k)\mathbf{H2}(k) \end{aligned} \tag{6}$$

where $\mathbf{H1}(k)=\mathbf{H1}(X(k))$ and $\mathbf{H2}(k)=\mathbf{H2}(X(k))$.

Remark 2. *Suppose that* $A = Y(k)^T$ *and* $B = W1(k)\boldsymbol{H1}(k)$*, then, we have*

$$Y(k)^T W1(k)\boldsymbol{H1}(k) = A_{1\times q}B_{q\times 1} = B^T A^T = (W1(k)\boldsymbol{H1}(k))^T Y(k).$$

We used this relation in (6).

To derive the SGD learning algorithm, the following relations are required:

$$\begin{aligned}\frac{\partial J(k)}{\partial W1(k)} &= \left(-Y(k)\mathbf{H1}(k)^T + W1(k)\mathbf{H1}(k)\mathbf{H1}(k)^T + W2(k)\mathbf{H2}(k)\mathbf{H1}(k)^T\right)\\ &= \left(-Y(k) + W1(k)\mathbf{H1}(k) + W2(k)\mathbf{H2}(k)\right)\mathbf{H1}(k)^T\\ &= -E(k)\mathbf{H1}(k)^T, \end{aligned} \tag{7}$$

$$\begin{aligned}\frac{\partial J(k)}{\partial W2(k)} &= \left(-Y(k)\mathbf{H2}(k)^T + W2(k)\mathbf{H2}(k)\mathbf{H2}(k)^T + W1(k)\mathbf{H1}(k)\mathbf{H2}(k)^T\right)\\ &= \left(-Y(k) + W2(k)\mathbf{H2}(k) + W1(k)\mathbf{H1}(k)\right)\mathbf{H2}(k)^T\\ &= -E(k)\mathbf{H2}(k)^T, \end{aligned} \tag{8}$$

From the Equations (7) and (8), we can conclude that

$$\Delta W1(k) = -\Gamma_1 E(k)\mathbf{H1}(k)^T \tag{9}$$

$$\Delta W2(k) = -\Gamma_2 E(k)\mathbf{H2}(k)^T \tag{10}$$

where the matrices Γ_1 and Γ_2 are the learning gains. In this work, we suppose that Γ_1 and Γ_2 are positive definite matrices.

Suppose that HONN using the ideal parameters $W1_\star$ and $W2_\star$ can model the actual output $Y(k)$ as follows:

$$Y(k) = W1_\star \mathbf{H1}(k) + W2_\star \mathbf{H2}(k) + \varepsilon(k).$$

The ideal parameters $W1_\star$ and $W2_\star$ are unknown, and therefore, they must be approximated. Suppose that $W1(k)$ and $W2(k)$ are approximations for $W1_\star$ and $W2_\star$, respectively. Therefore the output of HONN $\widehat{Y}$ can be written as

$$\widehat{Y}(k) = W1(k)\mathbf{H1}(k) + W2(k)\mathbf{H2}(k). \tag{11}$$

Then we may compute the model error as follows:

$$\begin{aligned} E(k) &= Y(k) - \widehat{Y}(k) = W1_\star \mathbf{H1}(k) + W2_\star \mathbf{H2}(k) + \varepsilon(k) - W1(k)\mathbf{H1}(k) - W2(k)\mathbf{H2}(k)\\ &= \widetilde{W}1(k)\mathbf{H1}(k) + \widetilde{W}2(k)\mathbf{H2}(k) + \varepsilon(k), \end{aligned} \tag{12}$$

where $\widetilde{W}1(k) = W1_\star - W1(k)$ and $\widetilde{W}2(k) = W2_\star - W2(k)$. The following theorem investigate the stability analysis of the proposed method.

Theorem 1. *Suppose that HONN is trained by the learning laws (9) and (10), and*

$$2E(k)^T\varepsilon(k) \leq (2-\beta)||E(k)||^2 \tag{13}$$

where Γ_1 *and* Γ_2 *are positive definite matrices and*

$$\beta = \lambda_{\max}(\Gamma_1)\kappa_1 + \lambda_{\max}(\Gamma_2)\kappa_2 \tag{14}$$

$$\kappa_1 = \max_k \|\boldsymbol{H1}(k)\|^2,\ \kappa_2 = \max_k \|\boldsymbol{H2}(k)\|^2 \tag{15}$$

Then, the error $E(k)$ *converges to zero as k tends to infinity.*

Proof. Consider the following Lyapunov function

$$v(k) = \mathrm{tr}\left(\widetilde{W}1(k)^T\Gamma_1^{-1}\widetilde{W}1(k)\right) + \mathrm{tr}\left(\widetilde{W}2(k)^T\Gamma_2^{-1}\widetilde{W}2(k)\right)$$

where $\widetilde{W}1(k) = W1_\star - W1(k)$, $\widetilde{W}2(k) = W2_\star - W2(k)$. At first, we notice that

$$\begin{aligned}
\mathrm{tr}\left(\widetilde{W}1_{i+1}^T\Gamma_1^{-1}\widetilde{W}1_{i+1}\right) &- \mathrm{tr}\left(\widetilde{W}1(k)^T\Gamma_1^{-1}\widetilde{W}1(k)\right) \\
&= \mathrm{tr}\left((\widetilde{W}1(k)+\Delta\widetilde{W}1(k))^T\Gamma_1^{-1}(\widetilde{W}1(k)+\Delta\widetilde{W}1(k))\right) - \mathrm{tr}\left(\widetilde{W}1(k)^T\Gamma_1^{-1}\widetilde{W}1(k)\right) \\
&= \mathrm{tr}\left(\widetilde{W}1(k)^T\Gamma_1^{-1}\widetilde{W}1(k)\right) + \mathrm{tr}\left(\Delta\widetilde{W}1(k)^T\Gamma_1^{-1}\widetilde{W}1(k)\right) \\
&\quad + \mathrm{tr}\left(\widetilde{W}1(k)^T\Gamma_1^{-1}\Delta\widetilde{W}1(k)\right) + \mathrm{tr}\left(\Delta\widetilde{W}1(k)^T\Gamma_1^{-1}\Delta\widetilde{W}1(k)\right) - \mathrm{tr}\left(\widetilde{W}1(k)^T\Gamma_1^{-1}\widetilde{W}1(k)\right) \\
&= 2\mathrm{tr}\left(\widetilde{W}1(k)^T\Gamma_1^{-1}\Delta\widetilde{W}1(k)\right) + \mathrm{tr}\left(\Delta\widetilde{W}1(k)^T\Gamma_1^{-1}\Delta\widetilde{W}1(k)\right)
\end{aligned} \tag{16}$$

Similar to (16), the other terms of $\Delta v(k)$ can be simplified. Therefore, we have

$$\begin{aligned}
\Delta v(k) &= v_{k+1} - v(k) \\
&= 2\mathrm{tr}\left(\widetilde{W}1(k)^T\Gamma_1^{-1}\Delta\widetilde{W}1(k)\right) + 2\mathrm{tr}\left(\widetilde{W}2(k)^T\Gamma_2^{-1}\Delta\widetilde{W}2(k)\right) \\
&\quad + \mathrm{tr}\left(\Delta\widetilde{W}1(k)^T\Gamma_1^{-1}\Delta\widetilde{W}1(k)\right) + \mathrm{tr}\left(\Delta\widetilde{W}2(k)^T\Gamma_2^{-1}\Delta\widetilde{W}2(k)\right) \\
&= -2\mathrm{tr}\left(\widetilde{W}1(k)^TE(k)\mathbf{H1}(k)^T\right) - 2\mathrm{tr}\left(\widetilde{W}2(k)^TE(k)\mathbf{H2}(k)^T\right) \\
&\quad + \mathrm{tr}\left(\mathbf{H1}(k)E(k)^T\Gamma_1E(k)\mathbf{H1}(k)^T\right) + \mathrm{tr}\left(\mathbf{H2}(k)E(k)^T\Gamma_2E(k)\mathbf{H2}(k)^T\right) \\
&= -2\mathrm{tr}\left(\widetilde{W}1(k)^T\mathbf{H1}(k)E(k)^T + \widetilde{W}2(k)^T\mathbf{H2}(k)E(k)^T\right) \\
&\quad + E(k)^T\Gamma_1E(k)\mathbf{H1}(k)^T\mathbf{H1}(k) + E(k)^T\Gamma_2E(k)\mathbf{H2}(k)^T\mathbf{H2}(k) \\
&= -2\mathrm{tr}\left(E(k)E(k)^T\right) + 2\mathrm{tr}\left(\varepsilon(k)E(k)^T\right) + E(k)^T\Gamma_1E(k)\|\mathbf{H1}(k)\|^2 \\
&\quad + E(k)^T\Gamma_2E(k)\|\mathbf{H2}(k)\|^2 \\
&\leq -2\|E(k)\|^2 + 2E(k)^T\varepsilon(k) + \lambda_{\max}(\Gamma_1)\|E(k)\|^2\|\mathbf{H1}(k)\|^2 + \lambda_{\max}(\Gamma_2)\|E(k)\|^2\|\mathbf{H2}(k)\|^2 \\
&\leq -2\|E(k)\|^2 + 2E(k)^T\varepsilon(k) + \left(\lambda_{\max}(\Gamma_1)\kappa_1 + \lambda_{\max}(\Gamma_2)\kappa_2\right)\|E(k)\|^2 \\
&= (\beta-2)\|E(k)\|^2 + 2E(k)^T\varepsilon(k).
\end{aligned} \tag{17}$$

According to the equation (13), we have: $\Delta v(k) < 0$. As a result, the sequence $(v(k))$ is decreasing and bounded below. Therefore, $(v(k))$ is convergent: $\lim_{i\to\infty} v(k) = v_\infty < \infty$. According to the equation (17), we have

$$0 < (2-\beta)\sum_{k=0}^{\infty} ||E(k)||^2 - 2\sum_{k=0}^{\infty} E(k)^T \varepsilon(k) = -\sum_{k=0}^{\infty} \Delta v(k) = v_0 - v_\infty < \infty. \tag{18}$$

Thus, $(E(k)) \in l^2$ and according to the Barbalat's lemma in discrete case, we have [18]: $\lim_{k\to\infty} E(k) = 0$, which completes the proof. □

Remark 3. *According to the equations (1) and (2), for $t \in [ih,(i+1)h)$, we have $0 \le \frac{1}{2h}(t-ih) < \frac{1}{2}$, and then,*

$$1 \le \cosh\left(\tfrac{1}{2h}(t-ih)\right) < \cosh\left(\tfrac{1}{2}\right) \approx 1.13,$$
$$0 \le \sinh\left(\tfrac{1}{2h}(t-ih)\right) < \sinh\left(\tfrac{1}{2}\right) \approx 0.52.$$

Therefore, for $i = 0,1,2,\ldots,m-1$, we have $H1_i(t) < 1.13$ and $H2_i < 0.52$. As a result, κ_1 and κ_2 are some constant numbers. By attention to the positive definiteness of Γ_1 and Γ_2, smaller values of β result in bigger values of $2-\beta$, which increases the probability of meeting the assumption (13). We can tune the learning rate matrices Γ_1 and Γ_2 for better model training. To have small values for β, we can choose the learning rates matrices with small eigenvalues.

Remark 4. *Recently, some works have been published in the context of identifying nonlinear systems using the neural networks where the Lyapunov stability theory (LST) is used to prove the error convergence [2–5, 19, 20, 23, 33]. In [33], the assumptions for stability proof contain the identification error, learning rates, activation functions, and their derivatives, and sometimes, the network parameters. In [2, 3, 5, 20], the assumptions for stability proof contain the unmodeled dynamics, the identification error, and some hyper-parameters of the algorithms where learning rates are not within them. In these works, for stability proof of the learning algorithms, the terms containing second order differences of parameters are ignored. This deficiency decreases the strength of stability proof. In [19], the extreme learning machines are trained with SGD-based learning algorithm. For the stability proof of this algorithm, the assumption is $0 < \lambda_{\max}(\Gamma_{SG}) < 2$, where Γ_{SG} is the learning rates matrices. It must be mentioned that in this work, the computations are done in the continuous-time framework that simplifies the justifications. In the present work, similar to the recent work [4], the assumption of stability proof is a combination of works mentioned above. The assumption (13) contains the unmodeled dynamics, the identification error, and the learning rates matrices.*

5 Syatem identification using HONN

In this Section, some benchmark nonlinear systems are identified with the proposed neural model HONN. To show the efficiency of proposed method, we compare it with sinusoidal neural network (SNN). For the implementation of proposed models, we consider the following form of discrete dynamic nonlinear systems:

$$Z(k) = f(Z(k-1), U(k-1)), \tag{19}$$

where $U(k)$ and $Z(k)$ are the vectors of system inputs and states at the time index k, respectively. Assume that this system is controllable and the Lipschitz's condition is satisfied for f.

To identify (19), the HONN is utilized in NARX (nonlinear autoregressive with exogenous input) configuration. In this configuration, the system inputs and outputs are fed to the model. Assume that the system (19) is completely controllable and the Lipschitzs condition is satisfied for the unknown function f. To identify (19), the proposed models are utilized in NARX (nonlinear autoregressive with exogenous input) configuration [26]. In this configuration, the system inputs and outputs are fed to the model. We can write the system (19) as $Z(k) = AZ(k-1) + g(Z(k-1); U(k-1))$ where the matrix A is Hurwitz and the function g is the nonlinear part of the system (19). Then, according to the equation (11), we approximate the function g with the proposed neural models. To implement the proposed model HONN, in the equations (1) and (2), we suppose that $T = 1$. In the following examples, we have $h = 1/m$ where m shows the number of orthogonal functions in **H1** and **H2**. Also, n_h denotes the number of hidden neurons in SNN.

5.1 A single input-single output nonlinear system

Consider the following discrete dynamic nonlinear system [25]:

$$z(k+1) = \frac{z(k)z(k-1)(z(k)+2.5)}{1+z(k)^2z(k-1)^2} + u(k) \tag{20}$$

The identification of (20) is done by HONN and sinusoidal neural network (SNN) [3]). The input signal is chosen as $u(k) = \sin(\frac{2\pi k}{25})$. The initial values of trainable parameters in SNN are uniformly distributed pseudorandom numbers between -2 and 2. The initial values of trainable parameters in HONN are uniformly distributed pseudorandom numbers between 0.05 and 0.05. The input vector of models is $x = [u(k-1), u(k-2), z(k-1), z(k-2), z(k-3), 1]^T$.

The algorithm design parameters for SNN are chosen as

$$A = 0.1,\ \ n_h = 10,\ 20,\ 30,\ 40,\ 50,\ \ \Gamma_1 = \Gamma_2 = 0.05 I_{n_h \times n_h},$$

where Γ_1 and Γ_2 are the learning rates matrices. The algorithm design parameters for HONN are chosen as

$$A = 0.1,\ \ h = 0.25,\ 0.2,\ 0.15,\ 0.1,\ 0.08,\ 0.05,\ \ \Gamma_1 = \Gamma_2 = 0.1 I_{n_h \times n_h},$$

where Γ_1 and Γ_2 are the learning rates matrices.

Figures 2 and 3 show the actual states, the estimated states, and the errors in the testing of SNN with 40 hidden neurons and HONN with $h = 0.2$ in the identification of (20), respectively.

From the Table 1 and the Figures 2 and 3, we can conclude that the performance of HONN in the identification of (20) is better than SNN. According to Table 1, decreasing the parameter h generally decreases the identification error wherein the SNN, sometimes, increasing the number of hidden neurons does not result in error decreasing.

Table 1: Performances comparison of SNN and HONN models in the identification of (20).

Model	n_h	h	Parameters	Train MSE	Test MSE
SNN	10	-	70	1.0399e-02	1.7896e-03
SNN	20	-	140	4.4874e-02	7.3730e-04
SNN	30	-	210	6.7156e-02	2.5798e-04
SNN	40	-	280	1.2087e-01	1.5307e-04
SNN	50	-	350	3.6409e-01	1.5869e-04
HONN	-	0.25	41	9.3136e-04	1.2567e-04
HONN	-	0.2	51	3.3819e-04	1.9484e-05
HONN	-	0.15	71	3.8505e-04	2.4753e-06
HONN	-	0.1	101	3.1583e-04	4.9561e-07
HONN	-	0.08	131	3.3863e-04	2.0623e-08
HONN	-	0.05	201	3.2136e-04	1.0487e-09

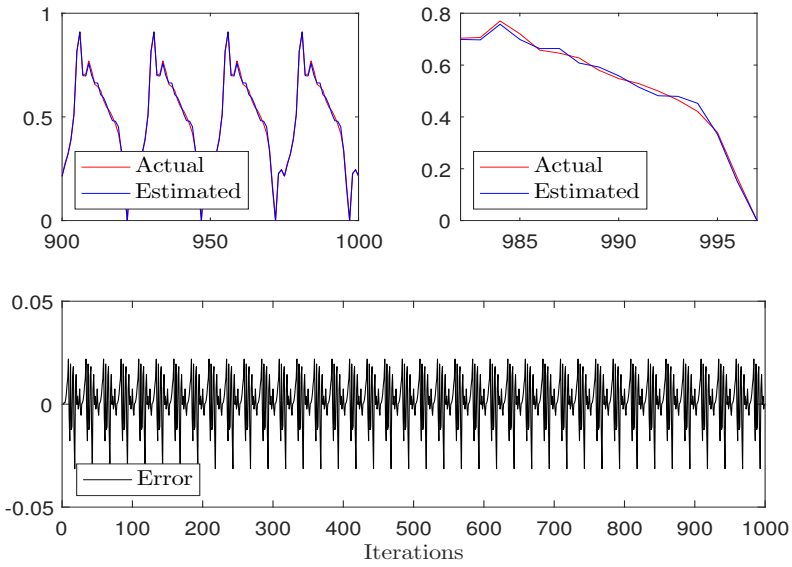

Figure 2: The actual states, the estimated states and the errors in the testing of SNN with 40 hidden neurons, in the identification of (20).

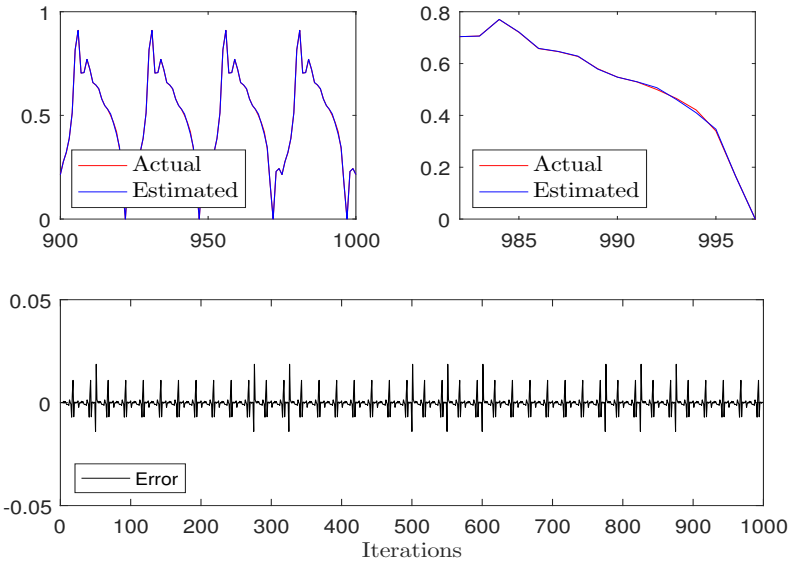

Figure 3: The actual states, the estimated states and the errors in testing of HONN with $h = 0.2$ (51 orthogonal functions), in the identification of system (20).

5.2 A multiple input-multiple output nonlinear system

Consider the following discrete dynamic nonlinear system [25]:

$$\begin{cases} z^1(k+1) = \frac{z^1(k)}{1+(z^2(k))^2} + u^1(k) \\ z^2(k+1) = \frac{z^1(k)z^2(k)}{1+(z^2(k))^2} + u^2(k), \quad z^1(0) = z^2(0) = 0. \end{cases} \tag{21}$$

The identification of (21) is done by HONN and SNN. The input signal is chosen as

$$u(k) = [\sin(\frac{2\pi k}{25}), \quad \cos(\frac{2\pi k}{25})]^T.$$

The initial values of trainable parameters in SNN are uniformly distributed pseudorandom numbers between -2 and 2. The initial values of trainable parameters in HONN are uniformly distributed pseudorandom numbers between 0.05 and 0.05. The input vector of models is $x = [u^1(k-1), u^2(k-1), z^1(k-1), z^2(k-1), 1]^T$.

The algorithm design parameters for SNN are chosen as

$$A = 0.1, \;\; n_h = 10,\; 20,\; 30,\; 40,\; 50,\; 60, \;\; \Gamma_1 = \Gamma_2 = 0.05 I_{n_h \times n_h},$$

where Γ_1 and Γ_2 are the learning rates matrices. The algorithm design parameters for HONN are chosen as

$$A = 0.1, \;\; h = 0.2,\; 0.15,\; 0.1,\; 0.08,\; 0.05, \;\; \Gamma_1 = \Gamma_2 = 0.1 I_{n_h \times n_h}, \tag{22}$$

Table 2: Performances comparison of SNN and HONN models in the identification of (21).

Model	n_h	h	Parameters	Train MSE	Test MSE
SNN	10	-	70	8.0354e-03	1.0431e-03
SNN	20	-	140	1.4402e-02	5.5962e-04
SNN	30	-	210	2.0616e-02	6.5447e-04
SNN	40	-	280	5.7140e-02	6.1418e-04
SNN	50	-	350	5.8536e-02	5.6590e-04
SNN	60	-	420	1.5363e-01	4.8967e-03
HONN	-	0.2	41	2.7960e-03	6.2670e-04
HONN	-	0.15	57	8.2662e-04	1.6329e-05
HONN	-	0.1	81	5.9234e-04	3.8162e-08
HONN	-	0.08	105	5.9180e-04	3.6184e-11
HONN	-	0.05	161	5.9857e-04	6.4671e-12

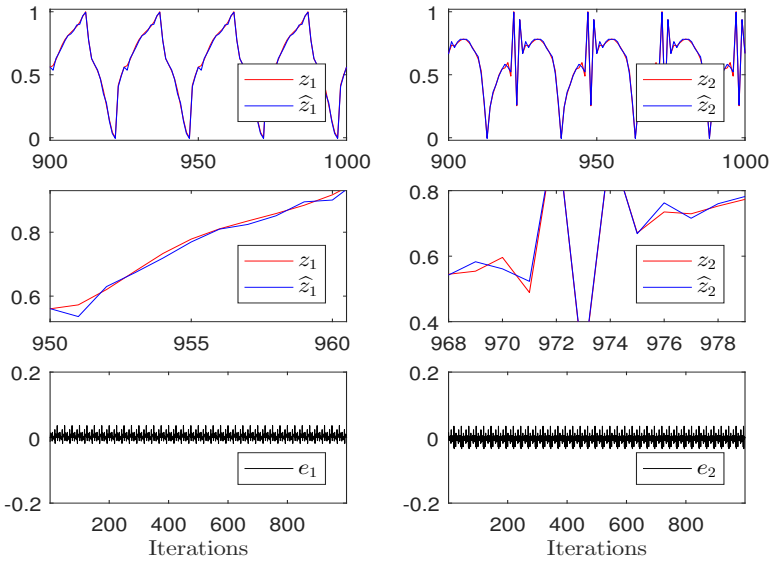

Figure 4: The actual states, the estimated states, and the errors in testing of SNN with 35 hidden neurons, in the identification of (21).

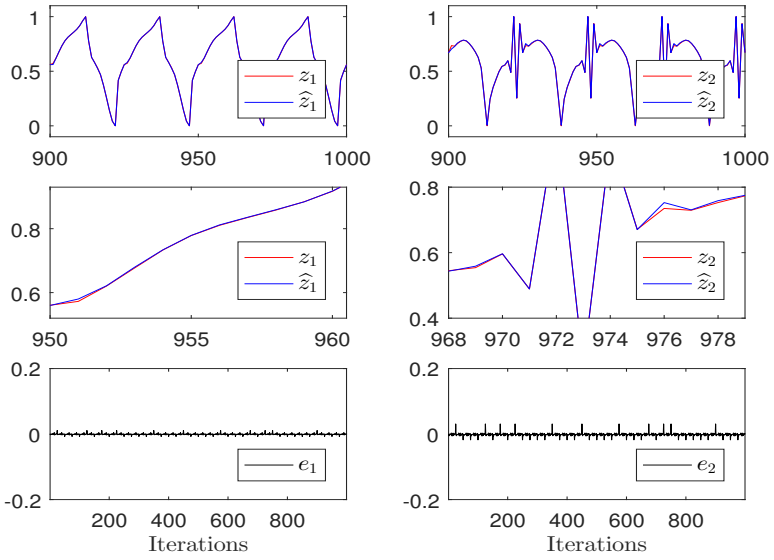

Figure 5: The Actual states, the estimated states, and the errors in testing of HONN with $h = 0.15$, in the identification of system (21).

where Γ_1 and Γ_2 are the learning rates matrices.

Figures 4 and 5 show the actual states, the estimated states, and the errors in the testing of SNN with 35 hidden neurons and HONN with $h = 0.15$, in the identification of (21), respectively.

From Table 2 and the Figures 4 and 5, we can conclude that in the presence of noise, the performance of HONN in the identification of (21) is better than SNN. According to Table 2, decreasing the parameter h generally decreases the identification error wherein the SNN, sometimes, increasing the number of hidden neurons does not result in error decreasing.

5.3 Discrete-Time Narendra-Li Benchmark System

Consider the discrete-time equations of the Narendra-Li system as follow [24]:

$$\begin{cases} z_1(k+1) &= \left(\frac{z_1(k)}{1+z_1(k)^2}+p(1)\right)\sin(z_2(k)), \\ z_2(k+1) &= z_2(k)\cos(z_2(k))+z_1(k)\exp\left(-\frac{z_1(k)^2+z_2(k)^2}{p(2)}\right) \\ & \quad +\frac{u(k)^3}{1+u(k)^2+p(3)\cos(z_1(k)+z_2(k))}, \end{cases} \tag{23}$$

$$y(k) = \frac{z_1(k)}{1+p(4)\sin(z_2(k))}+\frac{z_2(k)}{1+p(5)\sin(z_1(k))}, \tag{24}$$

where $z_1(k)$ and $z_2(k)$ are the states, $u(k)$ is the input signal, $y(k)$ is the output signal and p is a parameter vector with 5 elements. Here, we suppose that $p = [1.05,\ 7,\ 0.52,\ 0.52,\ 0.48]$.

The identification of (24) is done by HONN and SNN. The input signal is chosen as $u(k) = \sin(\frac{2\pi t}{10}) + \sin(\frac{2\pi t}{25})$. The initial values of trainable parameters in SNN are uniformly distributed pseudorandom numbers between -1 and 1. The initial values of trainable parameters in HONN are uniformly distributed

pseudorandom numbers between 0.05 and 0.05. The input vector of models is $x = [u(k-1), u(k-2), z(k-1), z(k-2), z(k-3), 1]^T$.

The algorithm design parameters for SNN are chosen as

$$A = 0.1, \;\; n_h = 10,\ 20,\ 30,\ 40,\ 50,\ 60, \;\; \Gamma_1 = \Gamma_2 = 0.01 I_{n_h \times n_h},$$

where Γ_1 and Γ_2 denote the learning rates. The algorithm design parameters for SHONN and HONN are chosen as

$$A = 0.1, \;\; h = 0.2,\ 0.15,\ 0.1,\ 0.085,\ 0.05, \;\; \Gamma_1 = \Gamma_2 = 0.1 I_{n_h \times n_h},$$

where Γ_1 and Γ_2 are the learning rates matrices.

Table 3: Performances comparison of SNN and HONN models in the identification of (24).

Model	n_h	h	Parameters	Train MSE	Test MSE
SNN	10	-	70	1.9062e-02	8.8108e-03
SNN	20	-	140	2.1483e-02	6.4271e-03
SNN	30	-	210	1.4921e-02	4.6987e-03
SNN	40	-	280	1.6315e-02	1.7501e-03
SNN	50	-	350	1.4665e-02	5.2370e-03
SNN	60	-	420	1.6637e-02	4.7391e-03
HONN	-	0.2	51	3.9229e-03	2.8217e-03
HONN	-	0.15	71	3.4466e-03	2.9197e-03
HONN	-	0.1	101	1.7831e-03	2.9128e-04
HONN	-	0.08	131	1.4785e-03	3.6393e-04
HONN	-	0.05	201	1.1905e-03	7.9203e-05

Figures 6 and 7 show the actual states, the estimated states, and the errors in the testing of SNN with 40 hidden neurons and HONN with $h = 0.05$, in the identification of (24), respectively.

From Table 3, and Figures 6 and 7, we can conclude that the performance of HONN in the identification of (24) is better than SNN. According to Table 3, decreasing the parameter h generally decreases the identification error wherein SNN, sometimes, increasing the number of hidden neurons does not result in error decreasing.

6 Conclusion

This work proposes a new orthogonal neural identifier using a set of hyperbolic piecewise continuous orthogonal functions. We train the proposed neural structure by a stochastic gradient descent algorithm, and prove its stability. The proposed methodology is accurate, and can cope with the problems of overfitting and trapping in local minima in traditional neural networks. Simulation results show the efficiencies of this approach. Future works focus on using this model for different problems in engineering and applied mathematics, such as system analysis, optimal control problems, and designing neural controllers.

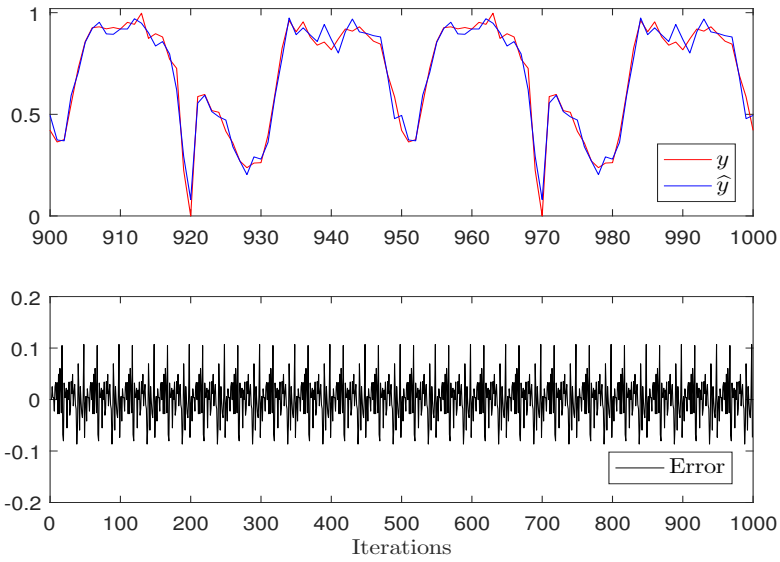

Figure 6: The actual states, the estimated states, and the errors in the testing of SNN with 40 hidden neurons, in the identification of (24).

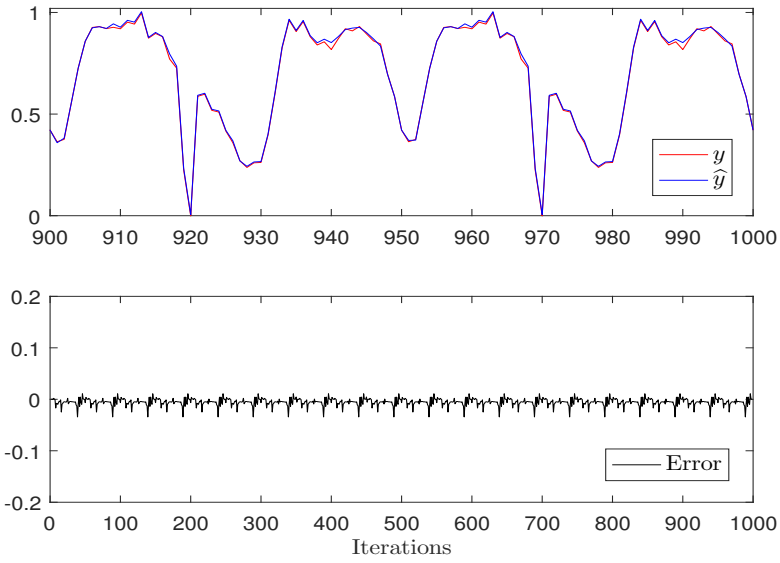

Figure 7: The actual states, the estimated states, and the errors in the testing of HONN with $h = 0.05$, in the identification of system (24).

References

[1] F. Abdollahi, A. Talebi, R. Patel, *Stable identification of nonlinear systems using neural networks: Theory and experiments*, IEEE/ASME Trans. Mechatron. **11** (2006) 488–495.

[2] G. Ahmadi, *Stable rough extreme learning machines for the identification of uncertain continuous-time nonlinear systems*, Control Optim. Appl. Math. **4** (2019) 83–101.

[3] G. Ahmadi, M. Teshnehlab, *Designing and implementation of stable sinusoidal rough-neural identifier*, IEEE Trans. Neural Netw. Learn. Syst. **28** (2017) 1774–1786.

[4] G. Ahmadi, M. Teshnehlab, *Identification of multiple input-multiple output non-linear system cement rotary kiln using stochastic gradient-based rough-neural network*, J. AI Data Min. **8** (2020) 417–425.

[5] G. Ahmadi, M. Teshnehlab, F. Soltanian, *A higher order online Lyapunov-based emotional learning for rough-neural identifiers*, *Control Optim. Appl. Math.* **3** (2018) 87–108.

[6] E. Babolian, A. Salimi Shamloo, *Numerical solution of Volterra integral and integro-differential equations of convolution type by using operational matrices of piecewise constant orthogonal functions*, J. Comput. Appl. Math. **214** (2008) 495-508.

[7] L. Bottou, *Stochastic gradient learning in neural networks*, in: Proceedings of Neuro-Nimes 91, EC2, Nimes, France, 1991.

[8] L. Bottou, *Large-scale machine learning with stochastic gradient descent*, in: Y. Lechevallier, G. Saporta (editors), *Proc. COMPSTAT2010*, 177-186, 2010, EC2, Nimes, France.

[9] S. Chen, S.A. Billings, P.M. Grant, *Non-linear systems identification using neural networks*, Research Report 370, Dept. of Automatic Control and System Engineering, University of Sheffield, 1989.

[10] K.B. Datta, B.M. Mohan, *Orthogonal Functions in Systems and Control*, World Scientific, Singapore, 1995.

[11] A. Deb, A. Dasguptab, G. Sarkar, *A new set of orthogonal functions and its application to the analysis of dynamic systems*, J. Franklin Inst. **343** (2006) 1-26.

[12] A. Deb, S. Roychoudhury, G. Sarkar, *Analysis and Identification of Time-Invariant Systems, Time-Varying Systems, and Multi-Delay Systems using Orthogonal Hybrid Functions: Theory and Algorithms with MATLAB*, Springer International Publishing, Switzerland, 2016.

[13] A. Deb, S. Roychoudhury, G. Sarkar, *Control System Analysis and Identification with MATLAB: Block Pulse and Related Orthogonal Functions*, Taylor & Francis Group, New York, 2019.

[14] M. Forgione, D. Piga, *Continuous-time system identification with neural networks: Model structures and fitting criteria*, Eur. J. Control **59** (2021) 69-81.

[15] M.H. Heydari, M. Razzaghi, *Piecewise Chebyshev cardinal functions: Application for constrained fractional optimal control problems*, Chaos Solitons Fractals **150**, 2021, 111118.

[16] S.M. Hoseini, H.R. Marzban,*Analysis of time-varying delay systems via triangular functions*, Appl. Math. Comput. **217** (2011) 74327441.

[17] S. Hykin, *Neural Networks: A Comprehensive Foundation*, Prentice Hall International, Canada, 1998.

[18] P. Ioannou, J. Sun, *Robust Adaptive Control*, Prentice Hall, New Jersey, 1996.

[19] V.M. Janakiraman, X. Nguyen, D. Assanis, *Stochastic gradient based extreme learning machines for stable online learning of advanced combustion engines*, Neurocomputing **177** (2016) 304-316.

[20] V.M. Janakiraman, X. Nguyen, D. Assanis, *A Lyapunov based stable online learning algorithm for nonlinear dynamical systems using extreme learning machines*, presented at the Int. Joint Conf. Neural Netw., Dallas, TX, Aug. 49, 2013.

[21] R. Kumar, S. Srivastava, A. Mohindru, *Lyapunov stability-dynamic back propagation-based comparative study of different types of functional link neural networks for the identification of nonlinear systems*, Soft Comput. **24** (2020) 5463–5482.

[22] T.T. Lee, J.T. Jeng, *The Chebyshev polynomial based unified model neural networks for function approximations*, IEEE Trans. Syst. Man Cybern. B **28** (1998) 925–935.

[23] Z. Man, H.R. Wu, S. Liu, X. Yu, *A new adaptive backpropagation algorithm based on Lyapunov stability theory for neural networks*,IEEE Trans. Neural Netw. **17** (2006) 1580–1591.

[24] K. S. Narendra, S.-M. Li., *Neural networks in control systems*, in: P. Smolensky, M. C. Mozer, D. E. Rumelhard, eds., Mathematical Perspectives on Neural Networks, 347-394, Lawrence Erlbaum Associates, Hillsdale, NJ, USA, 1996.

[25] K.S. Narendra, K. Parthasarathy, *Identification and control of dynamical systems using neural networks*, IEEE Trans. Neural Netw. **1** (1990) 4-27.

[26] O. Nelles, *Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models*, Springer-Verlag, Berlin, 2001.

[27] P. Netrapalli, *Stochastic gradient descent and its variants in machine learning*, J. Indian Inst. Sci. **99** (2019) 201–213.

[28] S. Purwar, I.N. Kar, A.N. Jha, *On-line system identification of complex systems using Chebyshev neural networks*, Appl. Soft Comput. **7** (2007) 364–372.

[29] H. Robbins, S. Monro, *A stochastic approximation model*, Ann. Math. Stat. **22** (1951) 400407.

[30] D.M. Sahoo, S. Chakraverty, *Functional link neural network approach to solve structural system identification problems*, Neural Comput. Appl. **30** (2018) 3327–3338.

[31] J. Tavoosi, A. Mohammadzadeh, K. Jermsittiparsert, *A review on type-2 fuzzy neural networks for system identification*, Soft Comput. **25** (2021) 7197–7212.

[32] S. Tomasiello, J.E. Macias-Diaz, A. Khastan, Alijani, *New sinusoidal basis functions and a neural network approach to solve nonlinear Volterra-Fredholm integral equations*, Neural Comput. Appl. **31** (2019) 4865-4878.

[33] W. Yu, *Nonlinear system identification using discrete-time recurrent neural networks with stable learning algorithms*, Inf. Sci. **158** (2004) 131147.

[34] N. Vukovic, M. Petrovic, Z. Miljkovic, *A comprehensive experimental evaluation of orthogonal polynomial expanded random vector functional link neural networks for regression*, Appl. Soft Comput. **70** (2018) 1083-1096.

[35] S.-S. Yang, C.-S. Tseng, *An orthogonal neural network for function approximation*, IEEE Trans. Syst. Man Cybern. B **26**(5) (1996) 779-784.