Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

High-Order Sliding Modes Based On-Line Training Algorithm for Recurrent High-Order Neural Networks

Alma Y. Alanis, * **Daniel Rios-Huerta,** * **Jorge D. Rios,** *
Nancy Arana-Daniel, * **Carlos Lopez-Franco** *
and Edgar N. Sanchez **

* *CUCEI, Universidad de Guadalajara, Guadalajara, Mexico, 44430 (e-mail: almayalanis@gmail.com).*
** *CINVESTAV, Unidad Guadalajara, Zapopan, Mexico, 45017*

Abstract: This work presents a discrete on-line training algorithm for recurrent high-order neural networks (RHONN). The proposed training algorithm is based on the arbitrary order differentiators of high-order sliding modes (HOSM) theory. Due to HOSM-based differentiators can approximate derivatives in finite time, the proposed training algorithm avoids the compute of the derivatives, unlike conventional training algorithms. The proposed HOSM-based algorithm is implemented for the training of a RHONN identifier, and its performance is compared with the results using the extended Kalman filter (EKF) training algorithm. Results of a implementation of the identifier for the Lorenz system and an implementation of the identifier for a tracked robot using experimental data are presented.

Keywords: Extended Kalman Filter, High order sliding mode, Neural identification, Neural network training, Robust exact differentiators.

1. INTRODUCTION

Artificial neural networks (ANNs) have become an important tool to solve several actual engineering problems, like pattern recognition, filtering or control. One of the main characteristics of ANNs is that they store knowledge in the synaptic weights values of the connections between their neurons. ANNs have the ability to learn and improve its performance through an interactive process of adjusting its synaptic weights by a set of well-defined rules called a learning algorithm Haykin (1998). Frequently, ANNs are trained with learning rules based on information from error derivatives with respect to the weights, such as the back-propagation (BP) algorithm that uses information from the first-order derivatives Werbos (1990), or the extended Kalman filter (EKF) based on the second-order derivatives information Puskorius and Feldkamp (2001). Due to the slow convergence of BP, a faster training algorithm emerged, based on EKF, which has been used in the last two decades to train ANNs Haykin (2001), Sanchez et al. (2008).

Recurrent neural networks are a type of ANN with one or more feedback loops. They are capable of having a state variables representation, therefore, they are suitable for different non-linear applications Haykin (1998). Recurrent high-order neural networks (RHONN) allow to approximate dynamic systems through high-order interactions between dynamical neurons Kosmatopoulos et al. (1995). There are several works where the RHONNs are trained with EKF to identify nonlinear systems Rios et al. (2013), Rios et al. (2015), Villaseñor et al. (2018). Nevertheless, the calculation of the derivatives is critical since the computational efficiency depends on it. In fact, for recurrent neural networks, the calculation of the derivatives using EKF requires high computational resources Puskorius and Feldkamp (2001). It is a drawback especially for real-time implementations. To avoid the derivatives calculation, in this work a training algorithm based on high-order sliding modes (HOSM) is proposed.

Sliding modes (SM) is a control technique that consists in designing a sliding variable of relative degree $r = 1$, that, with a discontinuous control action, the sliding variable is forced to zero in finite time and remains at zero even in the presence of perturbations and uncertainties Moreno (2018). Training algorithms based on continuous-time SM have been proposed in Poznyak et al. (1998), Poznyak et al. (1999), thus, they do not use the differentiators.

HOSM is an extension of basic sliding mode with an arbitrary degree $r > 1$, that mitigate the chattering effects which arise as high-frequency oscillations Moreno (2018), Utkin et al. (2009). HOSM implementation requires the r-th time derivatives of the sliding variables, which can be obtained with an observer called HOSM-based differentiator of order r-th, which is robust with respect to input noises and exact in their absence, as well as, it preserves the finite time convergence Shtessel et al. (2014). These HOSM-based differentiators are the base of the proposed training algorithm. Specifically, the differentiator presented in Levant and Livne (2018), taking advantage of a design parameter inclusion and its discretized representation that preserves the accuracy and robustness features

⋆ The authors thank the support of CONACYT Mexico, through Projects CB256769 and CB258068 ("Project supported by *Fondo Sectorial de Investigación para la Educación*")

of the continuous differentiator Levant (2014). Although, recently in Salgado and Chairez (2018) a continuous-time training algorithm based on the use of robust exact differentiator of the super twisting algorithm from the second-order sliding modes theory is proposed. In contrast, in the work we present, it is proposed a discrete-time training algorithm based on the robust exact differentiators with an arbitrary order.

On the other hand, most physical phenomena have a nonlinear behavior. The modeling of nonlinear systems is difficult due to its complexity, therefore, the identification of nonlinear systems is an alternative to obtained the model of a system Sanchez and Alanis (2006). The ANNs have been shown to be effective in the identification of nonlinear systems Narendra and Parthasarathy (1990). Therefore, to prove the performance of the proposed HOSM-based training algorithm, results using a RHONN identifier for the dynamic nonlinear Lorenz system and for an all-terrain tracked robot are presented. To show the performance obtained results are compared with the same RHONN on-line trained with EKF. In this way, the main contribution of this work is the presentation of the proposed HOSM training algorithm implemented for the training of RHONNs.

The outline of the work is the following, in Section 2 the mathematical preliminaries are described, first the discrete-time RHONN model, and then, the learning process to adjust the synaptic weights. In Section 3 the HOSM-based training algorithm is presented. Results are shown in Section 4, and finally, conclusions and future work are given in Section 5.

2. MATHEMATICAL PRELIMINARIES

2.1 Discrete-Time Recurrent High-Order Neural Network

Consider the following discrete-time RHONN described in Sanchez et al. (2008):

$$\hat{x}_i(k+1) = w_i^\top z_i(x(k), u(k)),\ i = 1, ..., p \tag{1}$$

where $\hat{x}_i$ is the state of the i-th neuron, w_i is the respective i-th online adapted weight vector of dimension $n \in [1, 2, ...)$, $x(k)$ is the plant state vector, $u = [u_1, u_1, ...u_m]^T$ is the input vector to the neural network, and $z_i(x(k), u(k))$ is given by

$$z_i(x(k), u(k)) = \begin{bmatrix} z_{i_1} \\ z_{i_2} \\ \vdots \\ z_{i_{L_i}} \end{bmatrix} = \begin{bmatrix} \prod_{j \epsilon I_1} \xi_{i_j}^{di_j(1)} \\ \prod_{j \epsilon I_2} \xi_{i_j}^{di_j(2)} \\ \vdots \\ \prod_{j \epsilon I_{L_i}} \xi_{i_j}^{di_j(L_i)} \end{bmatrix} \tag{2}$$

where L_i is the respective number of high-order connections, $\{I_1, I_2, ..., I_{L_i}\}$ is a collection of nonordered subsets of $\{1, 2, ..., p+m\}$, p is the state dimension, m is the number of external inputs, $di_j(k)$ being nonnegatives integers, and ξ_i is defined as follow:

$$\xi_i = \begin{bmatrix} \xi_{i_1} \\ \vdots \\ \xi_{i_p} \\ \xi_{i_{p+1}} \\ \vdots \\ \xi_{i_{p+m}} \end{bmatrix} = \begin{bmatrix} S(x_1) \\ \vdots \\ S(x_p) \\ u_1 \\ \vdots \\ u_m \end{bmatrix} \tag{3}$$

where the function $S(\bullet)$ is monotone-increasing, differentiable and is represented by sigmoids of the form:

$$S(\varsigma) = \frac{\alpha}{1 + e^{-\beta\varsigma}} - \gamma \tag{4}$$

where ς is any real valued variable.

2.2 Learning process

The adjustment of the synaptic weight $w_i(k)$ is determined by,

$$w_i(k+1) = w_i(k) + \Delta w_i(k) \tag{5}$$

where $w_i(k+1)$ is the adjusted weight, and in a supervised learning algorithm

$$\Delta w_i(k) = \eta f(e(k)) \tag{6}$$

is the adjustment term, where η is a positive constant known as learning-rate parameter, $e(k)$ is the error obtained through the difference between the desired output and the obtained output at the instant k, then, $f(e(k))$ is the function in terms of the error signal to be minimized Haykin (1998).

3. HIGH-ORDER SLIDING MODES BASED TRAINING ALGORITHM

3.1 High-order sliding modes differentiator

The differentiator designing problem is very important because numerical differentiation is found in many engineering applications Efimov and Fridman (2011), among these applications are the control theory Mboup et al. (2009), and the training of ANNs Salgado and Chairez (2018). The HOSM theory produces robust finite time-convergent exact differentiators Levant and Livne (2018), which are used for estimate the derivatives.

Hence, in this work, the development of a training algorithm for RHONNs based on a discrete version of the recursive differentiator (7) proposed in Levant and Livne (2018).

$$\begin{aligned} v_0 &= -\varphi_0(t, w_0 - e(t)) + w_1 \\ v_1 &= -\varphi_1(t, w_1 - v_0) + w_2 \\ &\vdots \\ v_n &= -\varphi_n(t, w_n - v_{n-1}) \end{aligned} \tag{7}$$

where φ_i is defined as:

$$\begin{aligned} \varphi_i(t, \delta(t)) = \lambda_{n-i} L^{\frac{1}{n-i+1}} |\delta(t)^{\frac{n-i}{n-i+1}}| \text{sign}(\delta(t)) \\ + M\mu_{n-i}\delta(t) \end{aligned} \tag{8}$$

where $L > 0$ is a Lipschitz constant, and for $\lambda_{n-i} > 1$, $\mu_{n-i} > 1$, there exist a positive double sequence $\{\lambda_i, \mu_i\}$,

$i = 0, 1, 2, ..., n$, such that for any n and in the absence of noise, the differentiator (7), (8) uniformly converges in finite time for any $M \geq 0$ Levant and Livne (2018). Moreover, it converges faster with large values of M. Parameters $\{\lambda_i,\ \mu_i\}$ are chosen recursively as in Levant (2003), some sequences are proposed in Levant and Livne (2018), Levant (2014).
Generally, the implementation of this differentiator is in a digital device, therefore, it is important to have a discrete-time representation. However, the accuracy of the differentiator (7) can be lost by the simplistic one-step Euler discretization. This problem is solved by the discretization presented in Levant (2014), Livne and Levant (2014), which through an approximation adding Taylor-like terms, preserves the accuracy of the continuous-time case.

3.2 High-order sliding modes training algorithm

The proposed HOSM training algorithm is base on the discretization of (7) presented in Levant (2014), such discretization is presented next:

$$\begin{aligned} w_0(k+1) &= w_0(k) - \varphi_0(k, w_0(k) - e(k))\tau \\ &\quad + \sum_{j=1}^{n} \frac{w_j(k)}{j!}\tau^j \\ &\vdots \\ w_i(k+1) &= w_i(k) - \varphi_i(k, w_i(k) - v_{i-1}(k))\tau \\ &\quad + \sum_{j=i+1}^{n} \frac{w_j(k)}{(j-i)!}\tau^{j-i} \\ &\vdots \\ w_n(k+1) &= w_n(k) - \varphi_n(k, w_n(k) - v_{n-1}(k))\tau \end{aligned} \tag{9}$$

where k is the sampling instants, and $\tau > 0$ is the constant sampling interval. If it is developed the first term of the summatory in each $w_i(k+1)$ equation in (9), it will result the term $w_{i+1}(k)$, then, if we regroup it with the term $-\varphi_i(k, w_i(k) - v_{i-1}(k))$, we will obtain the term $v_i(k)$ described in (7). Now it is possible to substitute (7) in (9) and results the following HOSM-based training algorithm:

$$\begin{aligned} w_0(k+1) &= w_0(k) + v_0(k)\tau + \sum_{j=2}^{n} \frac{w_j(k)}{j!}\tau^j \\ &\vdots \\ w_i(k+1) &= w_i(k) + v_i(k)\tau + \sum_{j=i+2}^{n} \frac{w_j(k)}{(j-i)!}\tau^{j-i} \\ &\vdots \\ w_n(k+1) &= w_n(k) + v_n(k)\tau \end{aligned} \tag{10}$$

Assumption: The training function of the RHONN is unknown and it is considered as the deterministic part of the error, then, it is assumed that is equal to zero since *a priori* information of both the system and the neural network is unknown, so that it is only preserved the stochastic term, therefore, $w_0(k) = w_0(k+1) = 0$. Otherwise, an offset error appears, and the error not reach the origin. Moreover, it is possible to add learning-rate parameters. Then, the following on-line HOSM training algorithm is proposed for the training of RHONNs:

$$\begin{aligned} v_0(k) &= -\varphi_0(t, -e(k)) + w_1(k) \\ v_1(k) &= -\varphi_1(t, w_1(k) - v_0(k)) + w_2(k) \\ w_1(k+1) &= w_1(k) + \eta\left[v_1(k)\tau + \sum_{j=3}^{n} \frac{w_j(k)}{(j-1)!}\tau^{j-1}\right] \\ &\vdots \\ v_i(k) &= -\varphi_i(t, w_i(k) - v_{i-1}(k)) + w_{i+1}(k) \\ w_i(k+1) &= w_i(k) + \eta\left[v_i(k)\tau + \sum_{j=i+2}^{n} \frac{w_j(k)}{(j-i)!}\tau^{j-i}\right] \\ &\vdots \\ v_n(k) &= -\varphi_n(t, w_n(k) - v_{n-1}(k)) \\ w_n(k+1) &= w_n(k) + \eta[v_n(k)\tau] \end{aligned} \tag{11}$$

In Fig. 1, the HOSM-based training algorithm is graphically described. It is possible to observe two inputs, the error as the training signal and the synaptic weights vector in the instant k, with dimension n. Thus, it is required an n-th order differentiator, where is not calculated $w_0(k+1)$ as already determined previously. The output is a weight vector adjusted with dimension n. Note the recursivity since $v_0(k)$ to $v_n(k)$ as in (7), which implies the importance of calculating $v_0(k)$.

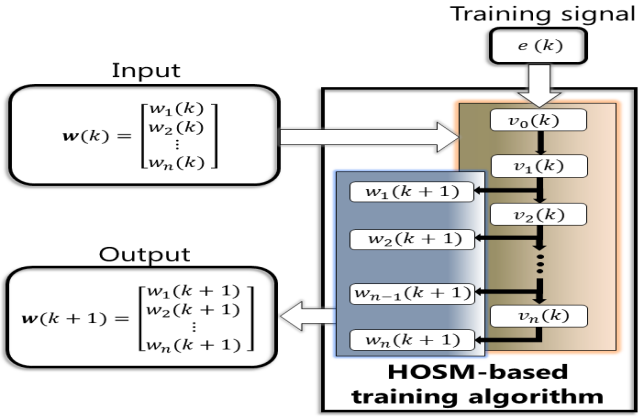

Fig. 1. HOSM-based training algorithm scheme.

4. RESULTS

4.1 Lorenz system RHONN identifier

To validate the performance of proposed training algorithm (11), it is implemented a RHONN as an identifier for the discrete-time Lorenz system Patel and Haykin (2001), which has a state of dimension three defined as (12),

$$\begin{aligned} x_1(k+1) &= x_1(k) + a[x_2(k) - x_1(k)]\tau \\ x_2(k+1) &= x_2(k) + [x_1(k)[r - x_3(k)] - x_2(k)]\tau \\ x_3(k+1) &= x_3(k) + [x_1(k)x_2(k) - bx_3(k)]\tau \end{aligned} \tag{12}$$

where $\tau > 0$ is the constant sampling interval, and $a = 10$, $b = 8/3$, and $r = 28$. Hence, it is considered that $p = 3$ in (1).
On the other hand, a synaptic weights vector of dimension $n = 4$ is chosen for each i-th neuron, such that the RHONN structure (1), without an input vector $u(k)$, it is designed heuristically as follows:

$$\begin{aligned}\hat{x}_1(k+1) &= w_{11}(k)S(x_3(k)) + w_{12}(k)S(x_1(k))^2 \\ &\quad + w_{13}(k)S(x_3(k)) + w_{14}(k)S(x_1(k))^2 \\ \hat{x}_2(k+1) &= w_{21}(k)S(x_1(k))^2 + w_{22}(k)S(x_3(k)) \\ &\quad + w_{23}(k)S(x_3(k)) + w_{24}(k)S(x_3(k)) \\ \hat{x}_3(k+1) &= w_{31}(k)S(x_3(k)) + w_{32}(k)S(x_1(k))^2 \\ &\quad + w_{33}(k)S(x_3(k)) + w_{34}(k)S(x_1(k))^2\end{aligned} \tag{13}$$

where $w_{ij}(k)$ is the j-th synaptic weight of the i-th neuron. As well by setting $\alpha = \beta = 2$ and $\gamma = 1$ as values for the activation function (4) to obtain the hyperbolic tangent as activation function, that is important to avoid saturation through input normalization Rovithakis and Christodoulou (2000).Considering the problem to approximate the Lorenz system (12) by (13), then, the state $\hat{x}_i$ of the i-th neuron corresponds to the i-th identified state variable of the Lorenz system Sanchez et al. (2008). Now, it is possible to define the error function $e(k)$ as follows:

$$e(k) = x(k) - \hat{x}(k) \tag{14}$$

equation (14) is the neural network training error.

4.2 RHONN identifier training

As from equations (13), note that is required to adjust four synaptic weights for each one of the three neurons. Hence, based in a fourth order differentiator the HOSM-based training algorithm (11), by expanding using (7) and (8) can be rewritten as the following equations:

$$\begin{aligned}\delta_0(k) &= -e(k) \\ v_0(k) &= -\lambda_4 L^{\frac{1}{5}} |\delta_0(k)^{\frac{4}{5}}| \text{sign}(\delta_0(k)) \\ &\quad - M\mu_4\delta_0(k) + w_1(k)\end{aligned} \tag{15}$$

$$\begin{aligned}\delta_1(k) &= w_1(k) - v_0(k) \\ v_1(k) &= -\lambda_3 L^{\frac{1}{4}} |\delta_1(k)^{\frac{3}{4}}| \text{sign}(\delta_1(k)) \\ &\quad - M\mu_3\delta_1(k) + w_2(k) \\ w_1(k+1) &= w_1(k) + \eta\left[v_1(k)\tau + \frac{1}{2!}w_3(k)\tau^2 \right. \\ &\quad \left. + \frac{1}{3!}w_4(k)\tau^3\right]\end{aligned} \tag{16}$$

$$\begin{aligned}\delta_2(k) &= w_2(k) - v_1(k) \\ v_2(k) &= -\lambda_2 L^{\frac{1}{3}} |\delta_2(k)^{\frac{2}{3}}| \text{sign}(\delta_2(k)) \\ &\quad - M\mu_2\delta_2(k) + w_3(k) \\ w_2(k+1) &= w_2(k) + \eta\left[v_2(k)\tau + \frac{1}{2!}w_4(k)\tau^2\right]\end{aligned} \tag{17}$$

$$\begin{aligned}\delta_3(k) &= w_3(k) - v_2(k) \\ v_3(k) &= -\lambda_1 L^{\frac{1}{2}} |\delta_3(k)^{\frac{1}{2}}| \text{sign}(\delta_3(k)) \\ &\quad - M\mu_1\delta_3(k) + w_4(k) \\ w_3(k+1) &= w_3(k) + \eta[v_3(k)\tau]\end{aligned} \tag{18}$$

$$\begin{aligned}\delta_4(k) &= w_4(k) - v_3(k) \\ v_4(k) &= -\lambda_0 L \text{sign}(\delta_4(k)) - M\mu_0\delta_4(k) \\ w_4(k+1) &= w_4(k) + \eta[v_4(k)\tau]\end{aligned} \tag{19}$$

where the double sequence $\{\lambda_i, \mu_i\}$ is chosen as in Levant (2014), i.e., $\{(1.1,1.5,2,3,5), (1,2,3,4,5)\}$, and by setting $\eta = 1$. The equations (15) to (19) are valid to train each one of i-th neurons. Besides, M is selected as 2.85 and L is selected as 1×10^{-6}.

4.3 Lorenz system: simulation results

Let the initial values of the state variables be $x_1(0) = 0.7061$, $x_2(0) = 0.5953$, and $x_2(0) = 0.7529$. In this simulation it is considered that in (13), the initial synaptic weights values are initialized as zero, $w_{ij}(0) = 0$, as well as, the initial values for the neural network state variables, $\hat{x}_1(0) = \hat{x}_2(0) = \hat{x}_3(0) = 0$. Nevertheless, all of them can be randomly selected and the accuracy will not be diminished.

The implementation is accomplished with a constant sampling time $\tau = 0.0001s$, and the number of samples used is $n_s = 600000$. Therefore, the simulation lasts $60s$. Moreover, for comparison purpose, the same RHONN identifier described in Section 4.1, is trained using EKF training algorithm Sanchez et al. (2008), with identical conditions. For spacing purposes, only the last 10s for each state variable is shown in Fig. 2, Fig. 3, and Fig. 4, respectively. Furthermore, Fig. 5 displays the phase portrait for the three systems: real, identified by HOSM, and identified by EKF. It is possible to see a similar performance with both HOSM and EKF training algorithms. However, the root-mean-square error (RMSE) and the mean absolute deviation (MAD) are calculated as a comparison criterion, results are presented in Table 1. We realize that the error obtained by the HOSM-based training algorithm is less than the obtained by EKF training algorithm only in the case of state variable x_2. Notwithstanding, we can claim that a RHONN can train with the HOSM-based training algorithm proposed and achieve good results.

Table 1. Identified Lorenz system error

	RMSE		MAD	
State variable	HOSM	EKF	HOSM	EKF
x_1	0.0067	0.0054	0.0033	0.0025
x_2	0.0082	0.0089	0.0046	0.0055
x_3	0.0059	0.0048	0.0016	0.0012

4.4 Neural identification of all-terrain tracked robot

As a second test for the proposed training algorithm, a modified HD2® Treaded All-Terrain Tracked Robot (ATR) is identified using the RHONN identifier (13),

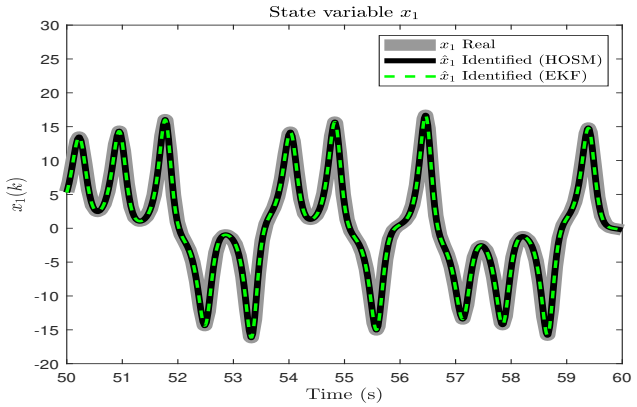

Fig. 2. State variable x_1 of Lorenz system. Thick line is the real measured state variable, thin line is the corresponding identified state variable using the HOSM algorithm, and the dashed line is the corresponding identified state variable using the EKF algorithm.

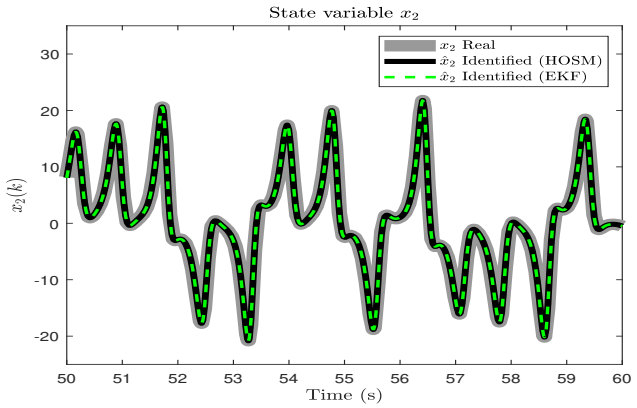

Fig. 3. State variable x_2 of Lorenz system. Thick line is the real measured state variable, thin line is the corresponding identified state variable using the HOSM algorithm, and the dashed line is the corresponding identified state variable using the EKF algorithm.

Table 2. Identified tracked robot error

	RMSE ($\times 10^{-3}$)		MAD ($\times 10^{-4}$)	
State variable	HOSM	EKF	HOSM	EKF
x_1	0.1466	0.1577	0.7015	0.7795
x_2	0.0497	0.0575	0.2406	0.3066
x_3	0.2060	0.1806	0.6545	0.5524

for this identification process real experimental data of tracked robot is used. For the training described in equations (15) - (19). Besides, M is selected as 3.9 and L is selected as 1×10^{-9}. Performance results are compared with the obtained results using EKF training algorithm, both results are shown in Table 2, and in Figs. 6, 7, where it is possible to see that the error with HOSM is less than EKF in the identification of x_1 and x_2 contrary to x_3.

5. CONCLUSIONS

In this work, a new training algorithm based on HOSM-differentiators for a RHONN is proposed. The proposed training algorithm is an online training that does not require to calculate the derivatives. A good performance in simulation of the proposed training algorithm can be seen through Figs. 2 to 5 and Table 1, as well as experimental results are shown in Figs. 6, 7 and Table 2. It is important to remark that the proposed training method is an online option which does not require the calculation of the derivatives, which can be especially computational beneficial for real-time implementations.

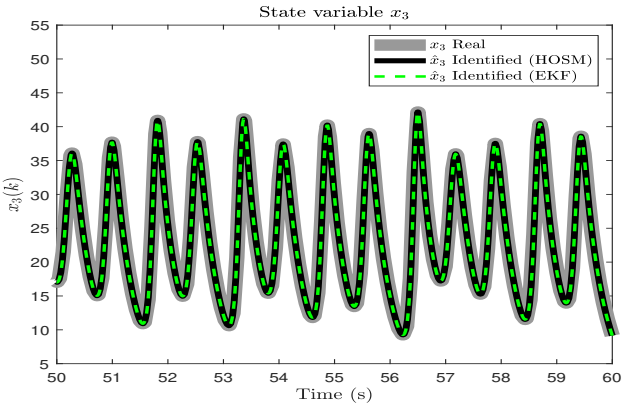

Fig. 4. State variable x_3 of Lorenz system. Thick line is the real measured state variable, thin line is the corresponding identified state variable using the HOSM algorithm, and the dashed line is the corresponding identified state variable using the EKF algorithm.

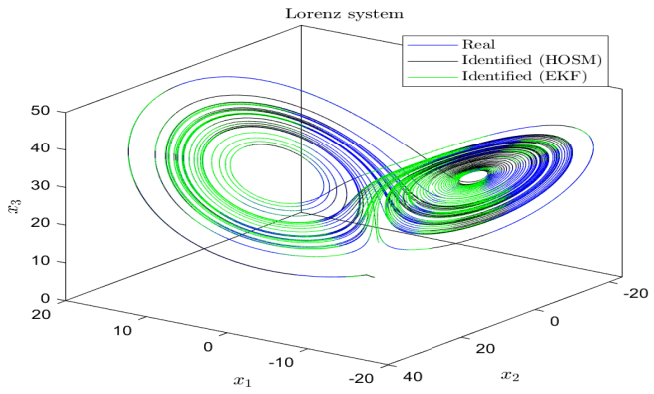

Fig. 5. Lorenz system phase portrait. Blue line is the real measured system, black line is the corresponding identified system using HOSM algorithm, and the green line is the corresponding identified system using EKF algorithm.

REFERENCES

Efimov, D.V. and Fridman, L. (2011). A hybrid robust non-homogeneous finite-time differentiator. *IEEE Transactions on Automatic Control*, 56(5), 1213–1219.

Haykin, S. (1998). *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 2nd edition.

Haykin, S. (2001). Kalman filters. In S. Haykin (ed.), *Kalman filtering and neural networks*, chapter 1, 1–21. John Wiley & Sons, Inc.

Kosmatopoulos, E., Polycarpou, M., and Christodoulou, M. (1995). High-order neural network structures for

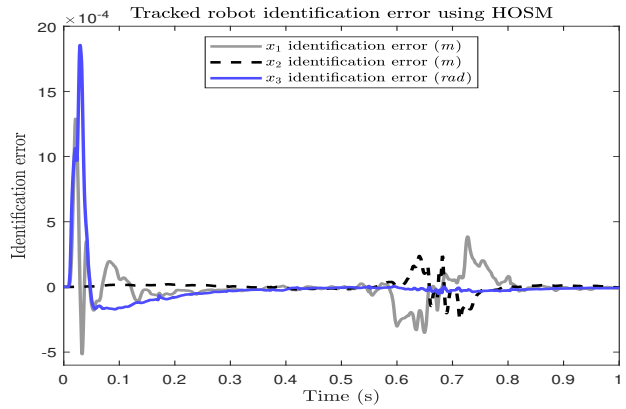

Fig. 6. Tracked robot identification error using HOSM training. The continuous gray line is the identification error of x_1, the dashed black line is the identification error of x_2, and the continuous blue line is the identification error of x_3.

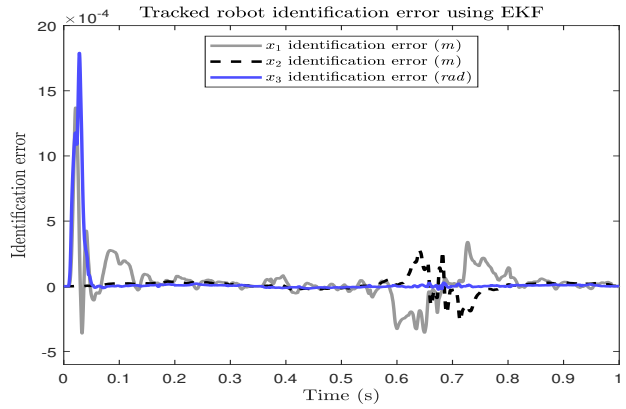

Fig. 7. Tracked robot identification error using EKF training. The continuous gray line is the identification error of x_1, the dashed black line is the identification error of x_2, and the continuous blue line is the identification error of x_3.

identification of dynamical systems. *IEEE Transactions on Neural Networks*, 6(1), 422–431.

Levant, A. (2014). Globally convergent fast exact differentiator with variable gains. In *2014 European Control Conference (ECC)*, 2925–2930.

Levant, A. (2003). Higher-order sliding modes, differentiation and output-feedback control. *International Journal of Control*, 76(9-10), 924–941.

Levant, A. and Livne, M. (2018). Globally convergent differentiators with variable gains. *International Journal of Control*, 91(9), 1994–2008.

Livne, M. and Levant, A. (2014). Proper discretization of homogeneous differentiators. *Automatica*, 50(8), 2007–2014.

Mboup, M., Join, C., and Fliess, M. (2009). Numerical differentiation with annihilators in noisy environment. *Numerical Algorithms*, 50(4), 439–467.

Moreno, J.A. (2018). Lyapunov-based design of homogeneous high-order sliding modes. In S. Li, X. Yu, L. Fridman, Z. Man, and X. Wang (eds.), *Advances in Variable Structure Systems and Sliding Mode Control—Theory and Applications*, chapter 1, 3–38. Springer International Publishing.

Narendra, K.S. and Parthasarathy, K. (1990). Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, 1(1), 4–27.

Patel, G.S. and Haykin, S. (2001). Chaotic dynamics. In S. Haykin (ed.), *Kalman filtering and neural networks*, chapter 4, 83–122. John Wiley & Sons, Inc.

Poznyak, A.S., Yu, W., Sanchez, E.N., and Sira-Ramirez, H. (1998). Robust identification by dynamic neural networks using sliding mode learning. *Applied Mathematics and Computer Science*, 8(1), 135–144.

Poznyak, A.S., Yu, W., and Sanchez, E.N. (1999). Identification and control of unknown chaotic systems via dynamic neural networks. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 46(12), 1491–1495.

Puskorius, G.V. and Feldkamp, L.A. (2001). Parameter-Based Kalman filter training: theory and implementation. In S. Haykin (ed.), *Kalman filtering and neural networks*, chapter 1, 1–21. John Wiley & Sons, Inc.

Rios, J.D., Alanis, A.Y., Arana-Daniel, N., and Lopez-Franco, C. (2015). RHONN identifier for unknown nonlinear discrete-time delay systems. In *2015 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)*, 1–5. doi: 10.1109/ROPEC.2015.7395076.

Rios, J.D., Alanis, A.Y., Rivera, J., and Hernandez-Gonzalez, M. (2013). Real-time discrete neural identifier for a linear induction motor using a dspace ds1104 board. In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, 1–6. doi: 10.1109/IJCNN.2013.6707109.

Rovithakis, G.A. and Christodoulou, M.A. (2000). *Adaptive Control with Recurrent High-order Neural Networks. Theory and Industrial Applications*. Springer-Verlag London.

Salgado, I. and Chairez, I. (2018). Adaptive unknown input estimation by sliding modes and differential neural network observer. *IEEE Transactions on Neural Networks and Learning Systems*, 29(8), 3499–3509.

Sanchez, E. and Alanis, A.Y. (2006). *Redes neuronales. Conceptos fundamentales y aplicaciones a control automático*. Pearson-Prentice Hall.

Sanchez, E.N., Alanis, A.Y., and Loukianov, A.G. (2008). *Discrete-Time High Order Neural Control. Trained with Kalman Filtering*. Springer-Verlag Berlin Heidelberg.

Shtessel, Y., Edwards, C., Fridman, L., and Levant, A. (2014). *Sliding Mode Control and Observation*. Springer New York.

Utkin, V., Guldner, J., and Shi, J. (2009). *Sliding mode control in electro-mechanical systems*. CRC Press.

Villaseñor, C., Rios, J.D., Arana-Daniel, N., Alanis, A.Y., Lopez-Franco, C., and Hernandez-Vargas, E.A. (2018). Germinal center optimization applied to neural inverse optimal control for an all-terrain tracked robot. *Applied Sciences*, 8(1), 1–31.

Werbos, P.J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10), 1550–1560. doi:10.1109/5.58337.