Research Article

Particle Swarm Based Approach of a Real-Time Discrete Neural Identifier for Linear Induction Motors

Alma Y. Alanis, E. Rangel, J. Rivera, N. Arana-Daniel, and C. Lopez-Franco

CUCEI, Universidad de Guadalajara, Apartado Postal 51-71, Col. Las Aguilas, Zapopan, Jalisco, Mexico

Correspondence should be addressed to Alma Y. Alanis; almayalanis@gmail.com

Received 3 July 2013; Accepted 1 December 2013

Academic Editor: Jun-Juh Yan

Copyright © 2013 Alma Y. Alanis et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper focusses on a discrete-time neural identifier applied to a linear induction motor (LIM) model, whose model is assumed to be unknown. This neural identifier is robust in presence of external and internal uncertainties. The proposed scheme is based on a discrete-time recurrent high-order neural network (RHONN) trained with a novel algorithm based on extended Kalman filter (EKF) and particle swarm optimization (PSO), using an online series-parallel configuration. Real-time results are included in order to illustrate the applicability of the proposed scheme.

1. Introduction

Linear induction motors (LIM) are a special electrical machine, in which the electrical energy is converted directly into mechanical energy of translatory motion. Strongest interest on these machines raised in the early 1970, however, in the late 1970, the research intensity and number of publications dropped. After 1980, LIM found their first noticeable applications in transportation industry, automation, and home appliances, among others [1, 2]. LIM has many excellent performance features such as high-starting thrust force, elimination of gears between motor and motion devices, reduction of mechanical losses and the size of motion devices, high speed operation, and silence [1, 3]. The driving principles of the LIM are similar to the traditional rotary induction motor (RIM), but its control characteristics are more complicated than the RIM, and the parameters are time varying due to the change of operating conditions, such as speed, temperature, and rail configuration.

Modern control systems usually require detailed knowledge about the system to be controlled; such knowledge should be represented in terms of differential or difference equations. This mathematical description of the dynamic system is named as the model. There can be different motives for establishing mathematical descriptions of dynamic systems, such as simulation, prediction, fault detection, and control system design. In this sense, basically there are two ways

to obtain a model; it can be derived in a deductive manner using physics laws, or it can be inferred from a set of data collected during a practical experiment. The first method can be simple, but in many cases it is excessively time-consuming; it would be unrealistic or impossible to obtain an accurate model in this way. The second method, which is commonly referred to as system identification [4], could be a useful shortcut for deriving mathematical models. Although system identification does not always result in an accurate model, a satisfactory one can be often obtained with reasonable efforts. The main drawback is the requirement to conduct a practical experiment, which brings the system through its range of operation [5].

Due to their nonlinear modeling characteristics, neural networks have been successfully applied in control systems, pattern classification, pattern recognition, and identification problems. The best well-known training approach for recurrent neural networks (RNN) is the back propagation through time [6]. However, it is a first-order gradient descent method, and hence its learning speed could be very slow. Another well-known training algorithm is the Levenberg-Marquardt one [6]; its principal disadvantage is that it can not guarantee the location of the global minimum and its learning speed could be slow too; this depends on the initialization. In the past years, extended Kalman filter (EKF) based algorithms have been introduced to train neural networks [7]. With

the EKF based algorithm, the learning convergence is improved [6]. The EKF training of neural networks, both feedforward and recurrent ones, has proven to be reliable for many applications [6]. However, EKF training requires the heuristic selection of some design parameters which is not always an easy task [7].

On the other hand particle swarm optimization (PSO) technique, which is based on the behavior of a flock of birds or school of fish, is a type of evolutionary computing technique [8]. It has been shown that the PSO training algorithm takes fewer computations and is faster than the BP algorithm for neural networks to achieve the same performance [8].

In this paper a recurrent high-order neural network (RHONN) is used to design the proposed neural identifier for nonlinear systems, whose mathematical model is assumed to be unknown. The learning algorithm for the RHONN is implemented using an extended Kalman filter with particle swarm optimization (EKF-PSO) based algorithm. In this paper a class of multiinput multioutput (MIMO) discrete-time nonlinear system is considered, for which a neural identifier is developed [9]; then this identifier is applied to a discrete-time unknown nonlinear system. The identifier is based on a recurrent high-order neural network (RHONN) [10], trained with an EKF-PSO based algorithm. The applicability of these schemes is illustrated experimentally for a linear induction motor (LIM).

The remainder of the paper is organized as follows. Section 2 is devoted to describing the neural model, based on the RHONN, in which the training phase relies on an extended Kalman filter which is able to deal with the nonlinearity of the model, and the improvement of EKF training algorithm based on a particle swarm optimization strategy. Section 3 reports the experimental results of the proposed method applied to the problem of identifying a three-phase induction motor. Finally Section 4 includes the conclusions and future work.

2. Preliminaries

Through this paper, k is used as the sampling step, $k \in \mathbb{N}$, $|\cdot|$ as the absolute value, and $\|\cdot\|$ as the Euclidian norm for vectors and as any adequate norm for matrices.

Consider a MIMO nonlinear system:

$$\begin{aligned}\chi(k+1) &= F(\chi(k), u(k)), \\ y(k) &= h(\chi(k)),\end{aligned}\quad (1)$$

where $\chi \in \mathcal{R}^n$, $u \in \mathcal{R}^m$, $y \in \mathcal{R}^p$, and finally, $F \in \mathcal{R}^n \times \mathcal{R}^m \rightarrow \mathcal{R}^n$ and $h \in \mathcal{R}^n \rightarrow \mathcal{R}^p$ are nonlinear functions.

2.1. Discrete-Time Recurrent High-Order Neural Networks. The use of multilayer neural networks is well known for pattern recognition and for modelling of static systems. The NN is trained to learn an input-output map. Theoretical works have proven that, even with just one hidden layer, a NN can uniformly approximate any continuous function over a compact domain, provided that the NN has a sufficient number of synaptic connections ([11, 12]).

For control tasks, extensions of the first-order Hopfield model called recurrent high-order neural networks (RHONN), which present more interactions among the neurons, are proposed in [13]. Additionally, the RHONN model is very flexible and allows us to incorporate to the neural model a priori information about the system structure. Besides, discrete-time neural networks are better fitted for real-time implementations [7].

Consider the following discrete-time recurrent high-order neural network (RHONN):

$$x_i(k+1) = w_i^\top z_i(x(k), u(k)), \quad i = 1, \dots, n, \quad (2)$$

where x_i ($i = 1, 2, \dots, n$) is the state of the i th neuron, L_i is the respective number of higher-order connections, $\{I_1, I_2, \dots, I_{L_i}\}$ is a collection of nonordered subsets of $\{1, 2, \dots, n\}$, n is the state dimension, w_i ($i = 1, 2, \dots, n$) is the respective online adapted weight vector, and $z_i(x(k), u(k))$ is given by

$$z_i(x(k), u(k)) = \begin{bmatrix} z_{i_1} \\ z_{i_2} \\ \vdots \\ z_{i_{L_i}} \end{bmatrix} = \begin{bmatrix} \prod_{j \in I_1} y_j^{d_{ij}(1)} \\ \prod_{j \in I_2} y_j^{d_{ij}(2)} \\ \vdots \\ \prod_{j \in I_{L_i}} y_j^{d_{ij}(L_i)} \end{bmatrix} \quad (3)$$

with $d_{ij}(k)$ being a nonnegative integers, and y_i is defined as follows:

$$y_i = \begin{bmatrix} y_{i_1} \\ \vdots \\ y_{i_n} \\ y_{i_{n+1}} \\ \vdots \\ y_{i_{n+m}} \end{bmatrix} = \begin{bmatrix} S(x_1) \\ \vdots \\ S(x_n) \\ u_1 \\ \vdots \\ u_m \end{bmatrix}. \quad (4)$$

In (4), $u = [u_1, u_2, \dots, u_m]^\top$ is the input vector to the neural network and $S(\cdot)$ is defined by

$$S(x) = \frac{1}{1 + \exp(-\beta x)}. \quad (5)$$

Consider the problem to approximate the i th plant state for the general discrete-time nonlinear system (1), by the following discrete-time RHONN [14]:

$$\chi_i(k+1) = w_i^{*\top} z_i(x(k), u(k)) + \epsilon_{z_i}, \quad i = 1, \dots, n, \quad (6)$$

where χ_i is the i th plant state, ϵ_{z_i} is a bounded approximation error, which can be reduced by increasing the number of the adjustable weights [13]. Assume that there exists ideal unknown weights vector w_i^* such that $\|\epsilon_{z_i}\|$ can be minimized on a compact set $\Omega_{z_i} \subset \mathcal{R}^{L_i}$. The ideal weight vector w_i^* is an artificial quantity required only for analytical purposes and is defined as

$$w_i^* = \arg \min_{w_i} \left\{ \sup_{\chi, u} |F_i(\chi(k), u(k)) - w_i^\top z_i(\cdot)| \right\}. \quad (7)$$

It is assumed to be unknown and it is the optimal set which renders the minimum approximation error, defined as ε_{z_i} ; $F_i(\cdot)$ is the i th component of $F(\cdot)$ [5, 13]. Let us define its estimate as w_i and the estimation error as

$$\tilde{w}_i(k) = w_i^* - w_i(k). \quad (8)$$

Due to this fact, we use $w_i(k)$ as the approximation of the weight vector w_i^* and ε_{z_i} , the modelling error, corresponds to $w_i^* \neq w_i(k)$. The estimate w_i is used for stability analysis which will be discussed later. Since w_i^* is constant, then $\tilde{w}_i(k+1) - \tilde{w}_i(k) = w_i(k+1) - w_i(k)$, for all $k \in \mathbb{N}$.

2.2. The EKF Training Algorithm. It is well-known that Kalman filtering (KF) estimates the state of a linear system with state and output additive white noises [15]. For KF-based neural network training, the network weights become the states to be estimated. In this case the error between the neural network output and the measured plant output can be considered as the additive white noise [6]. Although the white noise assumption is seldom satisfied, the developed algorithm has proven to be efficient in real applications [6, 7]. Due to the fact that the neural network mapping is nonlinear, an EKF-type of training is required [10]. The training goal is to find the weight values which minimize the prediction error. In this paper, an EKF-based training algorithm is used as follows:

$$\begin{aligned} w_i(k+1) &= w_i(k) + \eta_i K_i(k) e(k), \\ K_i(k) &= P_i(k) H_i(k) M_i(k), \\ P_i(k+1) &= P_i(k) - K_i(k) H_i^T(k) P_i(k) + Q_i(k), \\ i &= 1, \dots, n, \end{aligned} \quad (9)$$

with

$$\begin{aligned} M_i(k) &= [R_i(k) + H_i^T(k) P_i(k) H_i(k)]^{-1}, \\ e(k) &= y(k) - \hat{y}(k), \end{aligned} \quad (10)$$

where $e \in \mathbb{R}^p$ is the output estimation error and $P_i \in \mathbb{R}^{L_i \times L_i}$ is the weight estimation error covariance matrix at step k , $w_i \in \mathbb{R}^{L_i}$ is the weight (state) vector, L_i is the respective number of neural network weights, $y \in \mathbb{R}^p$ is the plant output, $\hat{y} \in \mathbb{R}^p$ is the NN output, n is the number of states, $K_i \in \mathbb{R}^{L_i \times p}$ is the Kalman gain matrix, $Q_i \in \mathbb{R}^{L_i \times L_i}$ is the NN weight estimation noise covariance matrix, $R_i \in \mathbb{R}^{p \times p}$ is the error noise covariance, and $H_i \in \mathbb{R}^{L_i \times p}$ is a matrix, in which each entry (H_{ij}) is the derivative of the i th neural output with respect to ij th neural network weight (w_{ij}), given as follows:

$$H_{ij}(k) = \left[\frac{\partial y(k)}{\partial w_{ij}(k)} \right]^T, \quad (11)$$

where $i = 1, \dots, n$ and $j = 1, \dots, L_i$. Usually P_i , Q_i , and R_i are initialized as diagonal matrices, with entries $P_i(0)$, $Q_i(0)$, and $R_i(0)$, respectively. Because typically the entries $P_i(0)$, $Q_i(0)$, and $R_i(0)$ are defined heuristically, in this paper, we propose the use of a PSO algorithm in order to compute online such entries to improve the EKF training algorithm, as follows.

2.3. PSO Improvement for EKF Training Algorithm. Particle swarm optimization (PSO) is a swarm intelligence technique developed by Kennedy and Eberhart in 1995 [16]. In fact, natural flocking and swarm behavior of birds and insects inspired them to develop the PSO algorithm. This technique has been used in several optimization and engineering problems ([8, 17–19]). In the basic PSO technique proposed by Kennedy and Eberhart [16], a great number of particles move around in a multidimensional space and each particle memorizes its position vector and velocity vector as well as the time at which the particle has acquired the best fitness. Furthermore, related particles can share data at the best-fitness time. The velocity of each particle is updated with the best positions acquired for all particles over iterations and the best positions are acquired by the related particles over generations [20].

To improve the performance of the basic PSO algorithm, some new versions of it have been proposed. At first, the concept of an inertia weight was developed to better control exploration and exploitation in [8, 20, 21]. Then, the research done by Clerc [22] indicated that using a constriction factor may be necessary to insure convergence of the particle swarm algorithm. After these two important modifications in the basic PSO were introduced, the multiphase particle swarm optimization (MPSO), the particle swarm optimization with Gaussian mutation, the quantum particle swarm optimization, a modified PSO with increasing inertia weight schedule, the Gaussian particle swarm optimization (GPSO), and the guaranteed convergence PSO (GCP SO) were introduced in [23].

In this paper the algorithm proposed in [8] is used in order to determine the design parameters for the EKF-learning algorithm. Initially a set of random solutions or a set of particles are considered. A random velocity is given to each particle and they are flown through the problem space. Each particle has memory which is used to keep track of the previous best position and corresponding fitness. The best value of the position of each individual is stored as p_{id} . In other words, p_{id} is the best position acquired by a particle during the course of its movement within the swarm. It has another value called the p_{gd} , which is the best value of all the particles p_{id} in the swarm. The basic concept of the PSO technique lies in accelerating each particle towards its p_{id} and p_{gd} locations at each time step. The PSO algorithm used in this paper is depicted in Figure 1 and can be defined as follows [8].

- (1) Initialize a population of particles with random positions and velocities in the problem space.
- (2) For each particle, evaluate the desired optimization fitness function.
- (3) Compare the particles fitness evaluation with the particles p_{id} ; if the current value is better than the p_{id} , then set p_{id} value equal to the current location.
- (4) Compare the best fitness evaluation with the population's overall previous best values. If the current value is better than the p_{gd} , then set p_{gd} to the particle's array and index value.

FIGURE 1: PSO algorithm used.

(5) Update the particle's velocity and position as follows.

The velocity of the i th particle of d dimension is given by

$$v_{id}(k+1) = c_0 v_{id}(k) + c_1 \text{rand}_1(p_{id}(k) - x_{id}(k)) + c_2 \text{rand}_2(p_{gd}(k) - x_{id}(k)). \quad (12)$$

The position vector of the i th particle of d dimension is updated as follows:

$$x_{id}(k+1) = x_{id}(k) + v_{id}(k), \quad (13)$$

where c_0 is the inertia weight, c_1 is the cognition acceleration constant, and c_2 is the social acceleration constant.

(6) Repeat step 2 until a criterion is met, usually a sufficiently good fitness or a maximum number of iterations or epochs.

In case that the velocity of the particle exceeds $V_{\max}$ (the maximum velocity for the particles) then it is reduced to $V_{\max}$. Thus, the resolution and fitness of search depends on the $V_{\max}$. If $V_{\max}$ is too high, then particles will move in larger steps and so the solution reached may not be as good as

expected. If $V_{\max}$ is too low, then particles will take a long time to reach the desired solution [8]. Because of the above explained PSO is a very suitable model to deal with noisy problems, as the one we are considering, besides that PSO has shown good results in optimization problems [8], it will be used to optimize the values for Kalman's filter covariance matrices instead of heuristic solutions. For this purpose, each particle will represent one of the Kalman's covariance entries.

2.4. Neural Identification. In the literature the capacity of RHONN to identify nonlinear systems in continuous-time [13] as well as in discrete-time [7, 24] is reported; however, the accurateness of results has a hard dependency on the RHONN structure selection as well as its training. For real-time implementations RHONN in discrete-time has shown good results; however, as it is demonstrated in [25], the identification error can be minimized increasing the number of high-order connections, although it is not possible to eliminate it for real-life problems. On the other hand, it is possible to reduce the identification error by means of the selection of an adequate training algorithm. The EKF training algorithm has proven to be reliable for many applications, particularly for real-time implementations [6, 24]. However, EKF training requires the heuristic selection of some design parameters which is not always an easy task; besides, the adequate selection of the design parameter affect directly the bound of the identification error [24]. Therefore a systematic methodology to select the design parameters is an important contribution for neural identification of unknown discrete-time nonlinear systems.

Now the RHONN (6) is trained with a EKF-PSO algorithm as defined above to identify the nonlinear system (2). First a RHONN structure is proposed; such structure can have physical significance or not; for control tasks it is better to consider a RHONN structure with physical significance. Then the EKF (9)-(10) is selected to implement the online series parallel training of the RHONN; however, as mentioned above, EKF training algorithm requires the adequate selection of many design parameters; particularly covariances matrices and learning rate affect directly the identification error and it is hard to determine their suitable values, in order to simplify the training algorithm tuning and to improve the identification process, the use of standard PSO algorithm (Figure 1) to determine such parameters is proposed in this paper. Finally an improved identification scheme for unknown discrete-time nonlinear systems is obtained.

3. Linear Induction Motor Application

In this section, the above developed scheme is applied to identify a three-phase linear induction motor. It is important to note that the proposed scheme is developed assuming that the plant model, parameters, and external disturbances (load torque) are unknown.

3.1. Motor Model. In order to illustrate the applicability of the proposed scheme, in this section, the proposed neural

identifier is applied to the α - β model of a LIM discretized by the Euler technique, which for the purposes of this paper is considered unknown, [26–28] as follows:

$$\begin{aligned}
 q_m(k+1) &= q_m(k) + v(k)T, \\
 v(k+1) &= (1 - K_2T)v(k) - k_1T\lambda_{r\alpha}(k)\rho_1i_{s\alpha}(k) \\
 &\quad - k_1T\lambda_{r\beta}(k)\rho_2i_{s\alpha}(k) + k_1T\lambda_{r\alpha}(k)\rho_2i_{s\beta}(k) \\
 &\quad - k_1T\lambda_{r\beta}(k)\rho_1i_{s\beta}(k) - k_3TF_L, \\
 \lambda_{r\alpha}(k+1) &= (1 - k_6T)\lambda_{r\alpha}(k) + k_4Tv(k)\rho_1i_{s\alpha}(k) \\
 &\quad - k_4T\rho_1i_{s\alpha}(k) + k_5T\rho_2i_{s\alpha}(k) + k_4T\rho_2i_{s\beta}(k) \\
 &\quad - k_4Tv(k)\rho_2i_{s\beta}(k) + k_5T\rho_1i_{s\beta}(k), \\
 \lambda_{r\beta}(k+1) &= (1 - k_6T)\lambda_{r\beta}(k) + k_4Tv(k)\rho_2i_{s\alpha}(k) \\
 &\quad - k_4T\rho_2i_{s\alpha}(k) - k_5T\rho_1i_{s\alpha}(k) + k_4T\rho_1i_{s\beta}(k) \\
 &\quad + k_4Tv(k)\rho_1i_{s\beta}(k) + k_5T\rho_2i_{s\beta}(k), \\
 i_{s\alpha}(k+1) &= (1 + k_9T)i_{s\alpha}(k) - k_7T\lambda_{r\alpha}(k)\rho_2 \\
 &\quad - k_8T\lambda_{r\alpha}(k)v(k)\rho_1 + k_7T\lambda_{r\beta}(k)\rho_1 \\
 &\quad - k_8T\lambda_{r\beta}(k)v(k)\rho_2 - k_{10}Tu_{\alpha}(k), \\
 i_{s\beta}(k+1) &= (1 + k_9T)i_{s\beta}(k) + k_8T\lambda_{r\alpha}(k)v(k)\rho_2 \\
 &\quad - k_7T\lambda_{r\alpha}(k)\rho_1 - k_7T\lambda_{r\beta}(k)\rho_2 \\
 &\quad - k_8T\lambda_{r\beta}(k)v(k)\rho_1 - k_{10}Tu_{\beta}(k)
 \end{aligned} \tag{14}$$

with

$$\begin{aligned}
 \rho_1 &= \sin(n_p q_m(k)), & \rho_2 &= \cos(n_p q_m(k)), \\
 k_1 &= \frac{n_p L_{sr}}{D_m L_r}, & k_2 &= \frac{R_m}{D_m}, \\
 k_3 &= \frac{1}{D_m}, & k_4 &= n_p L_{sr}, \\
 k_5 &= \frac{R_r L_{sr}}{L_r}, & k_6 &= \frac{R_r}{L_r}, \\
 k_7 &= \frac{L_{sr} R_r}{L_r (L_{sr}^2 - L_s L_r)}, & k_8 &= \frac{L_{sr} n_p}{L_{sr}^2 - L_s L_r}, \\
 k_9 &= \frac{L_r^2 R_s + L_{sr}^2 R_r}{L_r (L_{sr}^2 - L_s L_r)}, & k_{10} &= \frac{L_r}{L_{sr}^2 - L_s L_r},
 \end{aligned} \tag{15}$$

where q_m is the position, v is the linear velocity, $\lambda_{r\alpha}$ and $\lambda_{r\beta}$ are the α and β secondary flux components, respectively, $i_{s\alpha}$ and $i_{s\beta}$ are the α and β primary current components, respectively, $u_{s\alpha}$ and $u_{s\beta}$ are the α and β primary voltage components, respectively, R_s is the winding resistance per

phase, R_r is the secondary resistance per phase, L_{sr} is the magnetizing inductance per phase, L_s is the primary inductance per phase, L_r is the secondary inductance per phase, F_L is the load disturbance, D_m is the viscous friction and iron-loss coefficient, n_p is the number of poles pairs, and T is the sampling period [26].

It is important to note that this mathematical model is considered unknown for the design of the neural identifier. It is only included in this paper for completeness purposes.

3.2. Neural Identifier Design. The neural identifier proposed is designed as follows:

$$\begin{aligned}
 x_1(k+1) &= w_{11}(k)S(v(k)) + w_{12}(k)S(q_m(k)), \\
 x_2(k+1) &= w_{21}(k)S(v(k))^2 \\
 &\quad + w_{22}(k)S(\lambda_{r\alpha}(k))^2 + w_{23}(k)S(\lambda_{r\beta}(k))^{15}, \\
 x_3(k+1) &= w_{31}(k)S(v(k))^2 \\
 &\quad + w_{32}(k)S(\lambda_{r\alpha}(k))^2 + w_{33}(k)S(\lambda_{r\beta}(k))^{15}, \\
 x_4(k+1) &= w_{41}(k)S(v(k))^2 \\
 &\quad + w_{42}(k)S(\lambda_{r\alpha}(k))^2 + w_{43}(k)S(\lambda_{r\beta}(k))^{15}, \\
 x_5(k+1) &= w_{51}(k)S(v(k))^2 \\
 &\quad + w_{52}(k)S(\lambda_{r\alpha}(k))^2 + w_{53}(k)S(\lambda_{r\beta}(k))^2 \\
 &\quad + w_{54}(k)S(i_{s\alpha}(k))^3 + 0.02178u_{\alpha}(k), \\
 x_6(k+1) &= w_{61}(k)S(v(k))^2 \\
 &\quad + w_{62}(k)S(\lambda_{r\alpha}(k))^2 + w_{63}(k)S(\lambda_{r\beta}(k))^2 \\
 &\quad + w_{64}(k)S(i_{s\beta}(k))^3 + 0.02178u_{\beta}(k),
 \end{aligned} \tag{16}$$

where $S(x(k)) = \alpha \tan h(\beta x(k))$, $x_1(k)$ identifies $q_m(k)$, $x_2(k)$ identifies $v(k)$, $x_3(k)$ identifies $\lambda_{r\alpha}(k)$, $x_4(k)$ identifies $\lambda_{r\beta}(k)$, $x_5(k)$ identifies $i_{s\alpha}(k)$ and $x_6(k)$ identifies $i_{s\beta}(k)$. For this application only the fluxes are considered unmeasurable. The training is performed online, using a series-parallel configuration as shown in Figure 2. Both the NN and LIM states are initialized randomly. The associated covariances matrices are computed using the PSO algorithm and the RHONN weights are updated with the EKF as in (9). The input signals u_{α} and u_{β} are selected as chirp functions.

3.2.1. Reduced-Order Nonlinear Observer. The proposed neural identifier (16) requires the full state measurement assumption [26]. However, for real-time implementations, rotor fluxes measurement is a difficult task. Here, a reduced-order nonlinear observer is designed for fluxes on the basis of rotor

FIGURE 2: LIM identification scheme.

FIGURE 3: Schematic representation of the prototype to be identified.

speed and currents measurements. The flux dynamics in (14). Therefore, the following observer is used [29]:

$$\begin{aligned} \tilde{\Lambda}(k+1) = & \tilde{\Lambda}(k) - k_6 T \tilde{\Lambda}(k) - k_4 T \Theta^T J I_s(k) \\ & + k_4 T \Theta^T J I_s(k) v(k) + k_4 T \Theta^T I_s(k), \end{aligned} \quad (17)$$

where

$$\begin{aligned} \Lambda(k) &= \begin{bmatrix} \lambda_{r\alpha}(k) \\ \lambda_{r\beta}(k) \end{bmatrix}, \\ I_s(k) &= \begin{bmatrix} i_{s\alpha}(k) \\ i_{s\beta}(k) \end{bmatrix}, \\ \Theta(k) &= \begin{bmatrix} \cos(n_p q_m(k)) & -\sin(n_p q_m(k)) \\ \sin(n_p q_m(k)) & \cos(n_p q_m(k)) \end{bmatrix}, \\ J &= \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}. \end{aligned} \quad (18)$$

The stability proof for (17) is presented in [29].

3.3. Experimental Results. The proposed scheme is depicted in Figure 2. The experiments are performed using a benchmark whose schematic representation is depicted in Figure 3. Figure 4 shows the experimental benchmark for the LIM.

The methodology used to implement experimental identifier is as follows:

- (1) to validate and to test this algorithm via simulation in Matlab/Simulink, using a plant model and their respective parameters;
- (2) to download the validated identifier to the DS1104 board;
- (3) to replace the simulated model state variable values by the induction motor measurements current and angular position, acquired through the DS1104 board A/D ports, and calculated (fluxes) state variables values;

FIGURE 4: Linear induction motor prototype.

FIGURE 5: Position identification (zoom).

FIGURE 6: Linear velocity identification.

FIGURE 7: Alpha flux identification.

FIGURE 8: Beta flux identification.

FIGURE 9: Alpha current identification.

- (4) to send back through the DS1104 board the input signals (voltages) defined as chirp signals;
- (5) to process the input signals through the space vector pulse width modulation (SVPWM) power stage;
- (6) to apply the SVPWM output to the induction motor.

Experimental results are presented as follows. Figure 5 displays the detail for identification performance for position, it can be observed that there exists noise in the plant signal; however, RHONN is capable of identifying position signal. Figure 6 illustrates the identification performance for linear velocity; it is possible to note that identification errors remain bounded and decay faster to a minimum value; Figures 7 and 8 present the identification performance for

the fluxes in phase α and β , respectively, for both variables the identification is reached in least of 100 ms with an adequate accuracy. Figures 9 and 10 portray the identification performance for currents in phase α and β , respectively, from both figures. It is possible to appreciate that a suitable identification is performed; however, evolution of each one is different due to many circumstances as well as the structure selected for each neural state variable (16) and/or the presence of external and internal disturbances not necessarily equal for each component, among others. Finally Figure 11 shows the identification errors; it can be seen that all of them remain bounded. Besides, from Figure 11, it is possible to

FIGURE 10: Beta current identification.

TABLE 1: Mean value (μ) for identification error.

Variable	EKF	PSO-EKF
v	4.2271×10^{-4}	2.6469×10^{-4}
$\lambda_{r\alpha}$	6.2001×10^{-3}	2.8135×10^{-3}
$\lambda_{r\beta}$	1.5142×10^{-3}	4.2102×10^{-3}
i_{sa}	4.0230×10^{-1}	5.7903×10^{-3}
$i_{s\beta}$	2.7529×10^{-1}	5.5211×10^{-3}
q	9.0127×10^{-3}	1.7351×10^{-4}

note that the bounds for identification errors for each state variable are different in that they can be affected by internal and external disturbances of the system; in this particular experiment it can be observed that the beta components for current and flux are more affected than the alpha ones; this can be an instrumentation problem; however, in the case of neural network training, this problem can be seen as an opportunity field; due to the existence of internal and external disturbances the neural network can perform an adequate identification of the state variables and the inclusion of PSO improves such identification scheme as explained below. It is important to consider that experimental results, depicted in Figure 5 to Figure 11, have been obtained in open loop with chirp functions as inputs for LIM, in order to excite most of the plant dynamics; moreover, the neural states are initialized randomly.

3.4. Comparison of the EKF-PSO Algorithm for Neural Identification. In order to evaluate the performance of the EKF-PSO algorithm, it is compared with the typical EKF algorithm [24]; both training algorithms are used to identify a LIM in real time. The RHONN structures are exactly the same (16); only the training algorithm is changed in order to compare its performance.

Table 1 includes a comparison between the proposed PSO-EKF learning algorithm and the EKF one.

Results included in Table 1 show that the proposed methodology leads to an improvement of the results compared with respect to the EKF training algorithm.

FIGURE 11: Identification errors.

4. Conclusions

This paper has presented the application of recurrent high-order neural networks to the identification of discrete-time nonlinear systems. The training of the neural networks was performed online using an extended Kalman filter improved with a PSO algorithm. Experimental results illustrate the applicability of the proposed identification methodology for the online identification of a three-phase induction motor. The proposed neural identifier in our experiments was proved to be a model that captures very well the complexity for unknown discrete-time nonlinear systems. Finally, the use of PSO to improve the identification results has been experimentally illustrated in this paper.

Acknowledgments

The authors acknowledge the support of CONACYT Mexico, through Projects 103191Y, 106838Y, and 156567Y.

References

- [1] I. Boldea and S. A. Nasar, *Linear Electric Actuators and Generators*, Cambridge University Press, Cambridge, Mass, USA, 1997.
- [2] J. F. Gieras, *Linear Inductions Drives*, Oxford University Press, Oxford, UK, 1994.
- [3] I. Takahashi and Y. Ide, "Decoupling control of thrust and attractive force of a LIM using a space vector control inverter," *IEEE Transactions on Industry Applications*, vol. 29, no. 1, pp. 161–167, 1993.
- [4] W. Yu, "Nonlinear system identification using discrete-time recurrent neural networks with stable learning algorithms," *Information Sciences*, vol. 158, no. 1–4, pp. 131–147, 2004.
- [5] J. A. Farrell and M. M. Polycarpou, *Adaptive Approximation Based Control: Unifying Neural, Fuzzy and Traditional Adaptive Approximation Approaches*, John Wiley and Sons, New York, NY, USA, 2006.
- [6] S. Haykin, *Kalman Filtering and Neural Networks*, John Wiley and Sons, New York, NY, USA, 2001.
- [7] A. Y. Alanis, E. N. Sanchez, and A. G. Loukianov, "Discrete-time adaptive backstepping nonlinear control via high-order neural networks," *IEEE Transactions on Neural Networks*, vol. 18, no. 4, pp. 1185–1195, 2007.
- [8] R. Kiran, S. R. Jetty, and G. K. Venayagamoorthy, "Online training of a generalized neuron with particle swarm optimization," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN '06)*, pp. 5088–5095, Vancouver, Canada, July 2006.
- [9] A. S. Poznyak, E. N. Sanchez, and W. Yu, *Differential Neural Networks For Robust Nonlinear Control*, World Scientific, Singapore, 2001.
- [10] L. J. Ricalde and E. N. Sanchez, "Inverse optimal nonlinear recurrent high order neural observer," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN '05)*, pp. 361–365, Montreal, Canada, August 2005.
- [11] R. Chandra, M. Frean, and M. Zhang, "Adapting modularity during learning in cooperative co-evolutionary recurrent neural networks," *Soft Computing*, vol. 16, no. 6, pp. 1009–1020, 2011.
- [12] D. Richert, K. Masaud, and C. J. B. Macnab, "Discrete-time weight updates in neural-adaptive control," *Soft Computing*, vol. 17, no. 3, pp. 431–444, 2013.
- [13] G. A. Rovithakis and M. A. Chistodoulou, *Adaptive Control With Recurrent High-Order Neural Networks*, Springer, New York, NY, USA, 2000.
- [14] S. S. Ge, J. Zhang, and T. H. Lee, "Adaptive neural network control for a class of MIMO nonlinear systems with disturbances in discrete-time," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 34, no. 4, pp. 1630–1645, 2004.
- [15] R. Grover and P. Y. C. Hwang, *Introduction To Random Signals and Applied Kalman Filtering*, John Wiley and Sons, New York, NY, USA, 2nd edition, 1992.
- [16] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, December 1995.
- [17] W. Deng, R. Chen, B. He, Y. Liu, L. Yin, and J. Guo, "A novel two-stage hybrid swarm intelligence optimization algorithm and application," *Soft Computing*, vol. 16, no. 10, pp. 1707–1722, 2012.
- [18] Y. Liu and G. Pender, "Automatic calibration of a rapid flood spreading model using multiobjective optimisations," *Soft Computing*, vol. 17, no. 4, pp. 713–724, 2013.
- [19] R. L. Welch, S. M. Ruffing, and G. K. Venayagamoorthy, "Comparison of feedforward and feedback neural network architectures for short term wind speed prediction," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN '09)*, pp. 3335–3340, Atlanta, Ga, USA, June 2009.
- [20] S.-H. Zahiri and S.-A. Seyedin, "Swarm intelligence based classifiers," *Journal of the Franklin Institute*, vol. 344, no. 5, pp. 362–376, 2007.
- [21] Y. Shi and R. Eberhart, "Modified particle swarm optimizer," in *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC '98)*, pp. 69–73, May 1998.
- [22] M. Clerc, "The swarm and the queen: towards a deterministic and adaptive particle swarm optimization," *Proceedings of the Congress on Evolutionary Computation*, vol. 1, pp. 1951–1957.
- [23] B. Al-kazemi and C. K. Mohan, "Multi-phase generalization of the particle swarm optimization algorithm," *Proceedings of the Congress on Evolutionary Computation*, vol. 1, pp. 489–494, 2002.
- [24] A. Y. Alanis, E. N. Sanchez, A. G. Loukianov, and M. A. Perez-Cisneros, "Real-time discrete neural block control using sliding modes for electric induction motors," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 1, pp. 11–21, 2010.
- [25] E. B. Kosmatopoulos, M. M. Polycarpou, M. A. Christodoulou, and P. A. Ioannou, "High-order neural network structures for identification of dynamical systems," *IEEE Transactions on Neural Networks*, vol. 6, no. 2, pp. 422–431, 1995.
- [26] V. H. Benitez, A. G. Loukianov, and E. N. Sanchez, "Neural identification and control of a linear induction motor using an $\alpha = \beta$ model," in *Proceedings of the American Control Conference*, pp. 4041–4046, June 2003.
- [27] N. Kazantzis and C. Kravaris, "Time-discretization of nonlinear control systems via Taylor methods," *Computers and Chemical Engineering*, vol. 23, no. 6, pp. 763–784, 1999.
- [28] A. G. Loukianov, J. Rivera, and J. M. Cañedo, "Discrete time sliding mode control of an induction motor," in *Proceedings IFAC*, Barcelona, Spain, 2002.
- [29] M. Hernandez-Gonzalez, E. N. Sanchez, and A. G. Loukianov, "Discrete-time neural network control for a linear induction motor," in *Proceedings of the IEEE International Symposium on Intelligent Control (ISIC '08)*, pp. 1314–1319, September 2008.