Real-time Discrete Neural Identifier for a Linear Induction Motor using a dSPACE DS1104 Board

Jorge D. Rios, Alma Y. Alanis, Jorge Rivera, Miguel Hernandez-Gonzalez

Abstract—This paper presents a real-time discrete nonlinear neural identifier for a Linear Induction Motor (LIM). This identifier is based on a discrete-time recurrent high order neural network (RHONN) trained on-line with an extended Kalman filter (EKF)-based algorithm. A reduced order observer is used to estimate the secondary fluxes. The real-time implementation of the neural identifier is implemented by using dSPACE DS1104 controller board on MATLAB/Simulink with dSPACE RTI library and its performance is shown by graphs.

Keywords—Extended Kalman Filtering, Linear Induction Motor, Recurrent High Order Neural Networks, Real-time Neural Identification.

I. INTRODUCTION

Linear induction motors (LIMs) are special electrical machines, in which the electrical energy is converted directly into mechanical energy of translatory motion. Interest in these machines raised in the early 1970 [1]. In the late 1970, the research intensity and number of publications dropped. After 1980, LIMs found their first noticeable applications in, among others, transportation, industry, automation, and home appliances [1], [2]. LIMs have 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 motion device, high speed operation, silence and so on [2], [3]. The driving principles of the linear induction motor (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 [4].

On the other hand, modern control systems usually require a very structured 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 known as the model of the system. There can be several motives for establishing mathematical descriptions of dynamic systems, such as: simulation, prediction, fault detection, and control system design among others.

Basically, there are two ways to obtain a model; it can be derived in a deductive manner using laws of physics, 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 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 as system identification, could be a useful short cut for deriving mathematical models. Although system identification does not always result in an accurate model a satisfactory model can often be 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], [6].

Neural networks (NN) has grown to be a well-established methodology, which allows for solving very difficult problems in engineering, as exemplified by their applications to identification and control of general nonlinear and complex systems [7], [8], [9]. The neural networks, which involve dynamic elements in the form of feedback connections, are known as recurrent neural networks (RNN) [10].

Recurrent high order neural networks (RHONN) offer many advantages for modeling of complex nonlinear systems such as their excellent approximation capabilities and robustness against noise [7], [8], [9]. The main training approach for RNN is the back propagation through time learning which is a first order gradient descent method and it is known that its learning speed could be very slow [11], [12]. On the other hand, training methods based on Kalman filtering have been proposed in the literature [13], [14]. Due to the fact that training a neural network typically results in a nonlinear problem, the extended Kalman Filter is a common tool to use, instead of a linear Kalman filter [12], [14].

Extended Kalman filter (EKF) training for neural networks reduce the epoch size and the number of required neurons and the learning convergence is improved [12]. The EKF training of neural networks has proven to be reliable and practical [11].

The DS1104 R&D Controller Board is a board designed for the development of high-speed multivariable digital controllers and real-time simulations. Connector and panels provide access to input and output signals of the board. dSPACE RTI software library allows to communicate MATLAB/Simulink with the controller board, and provides functions to monitor the signals [15].

This paper presents the use of a recurrent high order neuronal network for real-time identification of a linear induction motor using a dSPACE DS1104 R&D controller board on MATLAB/Simulink with the dSPACE RTI software library. The training algorithm for the neural network is based on the extended Kalman filter [4], [13].
II. MATHEMATICAL PRELIMINARIES

A. Discrete-time high Order Neural Networks

Consider a MIMO nonlinear system:

\[ x(k+1) = F(x(k), u(k)) \quad (1) \]

where \( k \) is used as the step sampling, \( k \in 0 \cup \mathbb{Z}^+ \), \( x \in \mathbb{R}^n \), \( u \in \mathbb{R}^m \) and \( F \in \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n \) is a nonlinear function.

The use of multilayer neural networks is well known for pattern recognition and for modeling 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 [12].

For control task, extensions of the first order Hopfield model called Recurrent High Order Neural Networks (RHONN), which present more interactions among the neurons, are proposed in [16], [17]. Additionally, the RHONN model is very flexible and allows to incorporate to the neural model a priori information about the system structure [14].

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

\[ x_i(k+1) = \omega_i^T z_i(x(k), u(k)) \quad i = 1, \ldots, n \quad (2) \]

where \( x_i \) (\( i = 1,2,\ldots, n \)) is the state of the \( i \)-th neuron, \( \omega_i \) (\( i = 1,2,\ldots, n \)) is the respective number of adjustable weights [17]. The RHONN is very flexible and allows to incorporate to the neural model a priori information about the system structure [14].

Consider the problem to approximate the general nonlinear system (1), by the discrete-time RHONN series-parallel representation (6) [17], where \( \chi_i \) is the \( i \)-th plant state, \( \varepsilon_{\chi i} \) is a bounded approximation error, which can be reduced by increasing the number of adjustable weights [17]. Assume that there exists an ideal weight vector \( \omega_i^* \) such that the error \( \| \varepsilon_{\chi i} \| \) can be minimized on a compact set \( \Omega_{\chi i} \subset \mathbb{R}^L \). The ideal weight vector \( \omega_i^* \) is an artificial quantity required for analytical purpose [17]. In general, it is assumed that this vector exist and is constant but unknown.

B. The EKF Training Algorithm

The best well-known training approach for recurrent neural networks (RNN) is the back propagation through time learning [18]. However, it is a first order gradient descent learning method and hence its learning speed can be very slow [19]. Recently, extended Kalman filter (EKF) based algorithms have been introduced to train neural networks [20], [21]. With the EKF-based algorithm, the learning convergence is improved [18]. The EKF training of neural networks, both feedforward and recurrent ones, has proven to be reliable and practical for applications over the past eleven years [21].

It is known that Kalman filtering (KF) estimates the state of a linear system with additive state and output white noise [22], [23]. For EKF-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 additive white noise. Due to the fact that the neural network mapping is not linear, an EKF-type is required [24].

The training goal is to find the optimal weight values which minimize the predictions error. It is used an EKF-based training algorithm described by (7) and (8) [11].

\[ \alpha_i(k+1) = \alpha_i(k) + \eta_i K_i(k) e_i(k) \]

\[ K_i(k) = P_i(k) H_i(k) M_i(k) \]

\[ P_i(k+1) = P_i(k) - K_i(k) H_i^T(k) P_i(k) + Q_i(k) \quad i = 1,\ldots,n \]

\[ M_i(k) = [R_i(k) + H_i^T(k) P_i(k) H_i(k)]^{-1} \]

\[ e_i(k) = x_i(k) - \chi_i(k) \]

\[ H_i = \begin{bmatrix} \frac{\partial \chi_i(k)}{\partial \omega_i(k)} & \frac{\partial \chi_i(k)}{\partial \alpha_i(k)} \end{bmatrix}^T \quad (9) \]

where \( e_i(k) \in \mathbb{R} \) is the respective identification error, \( P_i(k) \in \mathbb{R}^{L_i \times L_i} \) is the weight estimation error covariance matrix at the step \( k \), \( \omega_i \in \mathbb{R}^{L_i} \) is the weight (state) vector, \( \chi_i \) is the \( i \)-th neural network state, \( K_i \in \mathbb{R}^{L_i} \) is the Kalman gain vector, \( Q_i \in \mathbb{R}^{L_i \times L_i} \) is the NN weight estimation noise
covariance, and \( R_t \in \mathbb{R} \) is the error noise covariance, and \( H_{ij} \in \mathbb{R}^{li} \) is a vector, in which each entry \( H_{ij} \) is the derivative of the neural network state \( (\chi_j) \), with respect to one neural network weight, \((\omega_{ij})\), given as (9) where \( i = 1, \ldots, n \) and \( j = 1, \ldots, l_i \). Usually \( P_t \) and \( Q_t \) are initialized as diagonal matrices, with entries \( P_t(0) \) and \( Q_t(0) \), respectively. It is important to remark that \( H_t(k) \), \( K_t(k) \) and \( P_t(k) \) for the EKF are bounded [23].

III. LINEAR INDUCTION MOTOR MODEL

The \( \alpha - \beta \) model of a LIM is discretized by the Euler technique [4], [25], [26] as follows

\[
q_a(k+1) = q_a(k) + v(k) T
\]
\[
v(k+1) = (1 - k_T T^o) q_a(k) - k_T \alpha_{a} (k) \rho_i a (k)
-k_T T^o \alpha_{a} (k) \rho_i a (k)
+k_T T^o \alpha_{a} (k) \rho_i a (k)
-k_T T^o \alpha_{a} (k) \rho_j a (k) - k_T T^o f_a
\]
\[
\lambda_{a}(k+1) = (1 - k_T T^o) \lambda_{a} (k) + k_T T^o \lambda_{a} (k)
-k_T T^o \lambda_{a} (k) + k_T T^o \lambda_{a} (k)
+k_T T^o \lambda_{a} (k) + k_T T^o \lambda_{a} (k)
-k_T T^o \lambda_{a} (k) + k_T T^o \lambda_{a} (k)
\]
\[
\lambda_{p} (k+1) = (1 - k_T T^o) \lambda_{p} (k) + k_T T^o \lambda_{p} (k)
-k_T T^o \lambda_{p} (k) + k_T T^o \lambda_{p} (k)
+k_T T^o \lambda_{p} (k) + k_T T^o \lambda_{p} (k)
-k_T T^o \lambda_{p} (k) + k_T T^o \lambda_{p} (k)
\]
\[
i_{a}(k+1) = (1 - k_T T^o) i_{a} (k) - k_T T^o i_{a} (k)
-k_T T^o i_{a} (k) - k_T T^o i_{a} (k)
+k_T T^o i_{a} (k) - k_T T^o i_{a} (k)
-k_T T^o i_{a} (k) - k_T T^o i_{a} (k)
\]
\[
i_{p} (k+1) = (1 - k_T T^o) i_{p} (k) + k_T T^o i_{p} (k)
-k_T T^o i_{p} (k) + k_T T^o i_{p} (k)
+k_T T^o i_{p} (k) - k_T T^o i_{p} (k)
-k_T T^o i_{p} (k) + k_T T^o i_{p} (k)
\]
\[
\rho_{a} = \sin(n_{a} \omega_{a}(k)) \quad \rho_{p} = \cos(n_{a} \omega_{a}(k))
\]

IV. FLUX OBSERVER

It is necessary to consider the fluxes cannot be measured. In order to overcome this problem, it is used a reduced order observer [13], [27]. The flux dynamics in (10) can be written as [28].

\[
\Psi(k+1) = \Psi(k) - k_T T^o \Psi(k) + k_T T^o I(k)
\]
\[
\Psi(k) = \begin{bmatrix}
\psi_{r_a}(k) \\
\psi_{r_p}(k)
\end{bmatrix}
J(k) = \begin{bmatrix}
J_{r_a}(k) \\
J_{r_p}(k)
\end{bmatrix}
\]

The observer proposed in [28] is (12)

\[
\Psi(k+1) = \Psi(k) - k_T T^o \Psi(k) + k_T T^o I(k)
\]
\[
\Psi(k) = \begin{bmatrix}
\psi_{r_a}(k) \\
\psi_{r_p}(k)
\end{bmatrix}
J(k) = \begin{bmatrix}
J_{r_a}(k) \\
J_{r_p}(k)
\end{bmatrix}
\]

The proof of convergence of this observer can be found in [13], [27], [28].

V. LIM RHONN IDENTIFIER

The proposed RHONN Identifier for the LIM model (10) is defined as:

\[
\chi_1(k+1) = \alpha_{1}(k) S(v(k)) + \alpha_{2}(k) S(\psi_{r_a}(k)) + \alpha_{3}(k) S(\psi_{r_p}(k))
\]
\[
\chi_2(k+1) = \alpha_{4}(k) S(\psi_{r_a}(k)) + \alpha_{5}(k) S(\psi_{r_p}(k))
\]
\[
\chi_3(k+1) = \alpha_{6}(k) S(\psi_{r_a}(k)) + \alpha_{7}(k) S(\psi_{r_p}(k))
\]
\[
\chi_4(k+1) = \alpha_{8}(k) S(\psi_{r_a}(k)) + \alpha_{9}(k) S(\psi_{r_p}(k))
\]

where \( q_a(k) \) is the position, \( v(k) \) is the linear velocity, \( \lambda_{a}(k) \) and \( \lambda_{p}(k) \) are the \( \alpha \)-axis and \( \beta \)-axis secondary flux, respectively, \( i_{a}(k) \) and \( i_{p}(k) \) are the \( \alpha \)-axis and \( \beta \)-axis primary current, respectively, \( u_{a}(k) \) and \( u_{p}(k) \) are the \( \alpha \)-axis and \( \beta \)-axis primary voltage, \( R_r \) is the winding resistance per phase, \( R_s \) is the secondary resistance per phase, \( L_m \) is the magnetizing inductance per phase, \( L_s \) is the primary inductance per phase, \( F_L \) is the load disturbance, \( D_m \) is the viscous friction and iron-loss coefficient and \( n_p \) is the number of poles pairs, \( T \) is the sample period [4].
where \( \chi_1(k) \) identifies the linear velocity, \( \chi_2(k) \) and \( \chi_3(k) \) identifies the \( \alpha \)-axis and \( \beta \)-axis secondary flux, respectively, \( \chi_4(k) \) and \( \chi_5(k) \) identifies the \( \alpha \)-axis and \( \beta \)-axis primary current, respectively. \( \chi_6(k) \) identifies the position.

The weights \( \omega_{ij}(k) \) are update by (7), \( S(x) \) is given by (S), the weights \( \omega_{14}(k), \omega_{15}(k), \omega_{24}(k), \omega_{25}(k), \omega_{34}(k), \omega_{35}(k), \omega_{45}(k) \) and \( \omega_{55}(k) \) remain fixed, more over \( \omega_{24}(k) = \omega_{25}(k) = \omega_{34}(k) = \omega_{35}(k) = \omega_f \), where \( \omega_f \) is a constant.

It is important to note that for identifier (13) the mathematical model (10) is considered unknown.

### VI. Real-time Test

For the real-time test the model shown in Fig. 1 is built in MATLAB/Simulink, and then implemented in dSPACE board using the software interface, the LIM model used is LAB – Volt® 8228 (Fig. 2). The inputs to the LIM are a frequency-variant signal given by the autotransformer (Fig. 3) and a PWM control coupled signal through a dSPACE DS1104 Board (Fig. 4) and RTI 1104 Connector Panel (Fig. 5). Both signals go through a power module shown (Fig. 6) which allows managing needed voltages and currents to move the secondary of the LIM.

\( \alpha \)-axis and \( \beta \)-axis currents are obtained from the three actual LIM currents converted to the \( \alpha-\beta \) model [4], position and velocity are obtained from the precision linear encoder SENC 150 mounted on the LIM.

The fixed weights values are: \( \omega_{14} = 0.001, \omega_{15} = 0.001, \omega_f = 0.001, \omega_{45} = 0.02178 \) y \( \omega_{55} = 0.02178 \).

The values of position, velocity and currents are the input parameters to observer which calculates \( \alpha \) – flux and \( \beta \) – flux. Afterwards, position, velocity, currents and fluxes are the inputs to the identifier which makes use of the EKF-based training algorithm (7), (8), (9) in order to adjust its weights \( \omega_{ij}(k) \) at each sampling time \( k \).

Figs. 7 to 12 show the identification errors between the neural identifier (13) and the LIM for a real-time test with a sampling time of 0.3s and a total time of 11s. Table 1 shows the root mean square errors (RMSE) between the neural identifier (13) and the LIM state signals.

<table>
<thead>
<tr>
<th>Identification</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Velocity</td>
<td>0.0089</td>
</tr>
<tr>
<td>( \alpha )-flux</td>
<td>5.2903e-005</td>
</tr>
<tr>
<td>( \beta )-flux</td>
<td>8.3943e-005</td>
</tr>
<tr>
<td>( \alpha ) current</td>
<td>0.2063</td>
</tr>
<tr>
<td>( \beta ) current</td>
<td>0.1657</td>
</tr>
<tr>
<td>Position</td>
<td>1.4944e-005</td>
</tr>
</tbody>
</table>

Fig. 1. MATLAB/Simulink Model.

Fig. 2. Linear Induction Motor LAB – Volt® 8228.

Fig. 3. Autotransformer.

Fig. 4. dSPACE DS1104 Controller Board.
Fig. 5. RTI 1104 Connector Panel

Fig. 6. Power Module.

Fig. 7. Velocity identification error.

Fig. 8. $\alpha$-flux identification error.

Fig. 9. $\beta$-flux identification error.

Fig. 10. $\alpha$ current identification error.

Fig. 11. $\beta$ current identification error.

Fig. 12. Position identification error.
VII. CONCLUSION

This paper proposes the use of RHONN for real-time identification of a LIM. The proposed RHONN identifier is trained on-line using an EKF-based algorithm. The proposed identifier is implemented on real-time using a dSPACE DS1104 Board, showing its applicability. Currently, the authors are working on improving the neural identifier performance and developing and identifier controller scheme.

ACKNOWLEDGMENT

The authors thank the support of CONACYT Mexico, through Project 103191Y.

REFERENCES