Current Controller for Induction Motor using an Artificial Neural Network trained with a Lyapunov based Algorithm

Julio Viola[a,b], José Restrepo[a,b,c], José Aller[a,b,c]
[a]Prometeo Project Researcher
[b]Universidad Politécnica Salesiana
Cuenca, Ecuador
{jcviola, restrepo}@ieee.org
[c]Universidad Simón Bolívar
Caracas, Venezuela
jaller@usb.ve

***Abstract*—This paper presents the use of a training algorithm based on a Lyapunov function approach applied to a stator current controller based on a state variable description of the induction machine plus a reference model. The results obtained with the proposed controller are compared with a previously reported method based on a Nonlinear Auto-Regressive Moving Average with eXogenous inputs (NARMAX) description of the induction machine. The proposed Lyapunov based training algorithm is used to ensure convergence of the weights towards a global minimum in the error function. Real time simulations employing a DSP based test bench are used to test the validity of the algorithms and the results are verified by a practical implementation of these controllers.**

***Keywords*—Backpropagation, Induction motor drives, Lyapunov methods, Neural Networks.**

I. Introduction

Nowadays, the three-phase induction motor is the most widely used machine in industrial applications. This is due mainly to its ruggedness, simpler structure, lower inertia and lower cost when compared with DC electric motors in the same power range. On the other hand it has the drawback of a non-linear torque speed relationship, due to the required slip needed to induce torque producing currents in the rotor. Therefore most of the control schemes developed for the induction motor are speed controls which, in turn, imply the control of electric torque. The electric torque can be controlled by controlling the modulus and phase of the stator currents, resulting in various control schemes known as vector controls [1]. The desired currents can be obtained by calculating and applying the appropriate voltage to the stator windings. A typical drive for the induction motor is a three leg voltage inverter, using IGBTs as switching devices and operating at constant switching frequency, which allows control of the mean stator voltage value by using pulse width modulation. Since highly non-linear relationships exist between stator voltages and stator currents, a non-linear current controller is required to obtain the stator voltages needed in each control cycle. In addition the induction machine typically has time-varying parameters, requiring the current controller to be adaptive, in order to obtain appropriate tracking of the stator current reference [2], [3]. Continuously online trained Artificial Neural Networks (ANN) are a natural choice for a non-linear adaptive controller thanks to their ability to map non-linear input-output relationships. Several control schemes have been proposed to control induction machines and the ability to learn and adapt to different machines have been analyzed [4],[5],[6],[7]. Previous works do not address, however, the issue of evolution of ANN weights in long tests, where a persistent drift can be observed in the weights value which can lead to instability in the ANN behaviour [8]. In [9] a general method is proposed for training ANNs based on a Lyapunov function approach which ensures convergence for the ANN weights. A practical application of this method has been reported in [10] where a single-phase synchronous rectifier is controlled with an ANN based current loop. In this paper a Model Reference Adaptive Control (MRAC) scheme based on a state variable model of the induction machine is used to define the structure of a neurocontroller for the current loop, and the results are compared with an equivalent current loop controller based on the NARMAX model of the machine. Both neurocontrollers are trained using a Lyapunov based algorithm and long term stability for the ANN weights it is demonstrated for the MRAC based controller. The paper is organized as follows: section II explains the proposed scheme used to control the induction machine, section III presents the expressions obtained for the Lyapunov training algorithm, section IV shows simulations of the control schemes and finally section V shows the results obtained with an experimental test rig.

II. Current controller

The stator current in the machine can be impressed by identifying its dynamics and finding the necessary stator voltage that produces the desired current. In this work artificial neural networks are used to identify the plant, first, as shown in Fig. 1, by using a NARMAX model of the machine, trained using a Lyapunov based algorithm as described in [5], and later by using a model reference and a state variable description of the machine, shown in Fig. 2. In the first case the stator current

978-1-4673-7553-5/15/$31.00 ©2015 IEEE

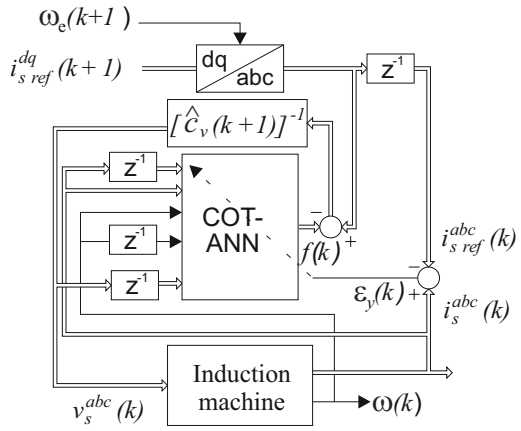

Figure 1: Model for the NARMAX based neurocontroller in natural a, b, c coordinates.

is described by the following expression [11]

$$\mathbf{i}_s(k+1) = \mathbf{f}\left(\mathbf{i}_s(k), \mathbf{i}_s(k-1), \omega_r(k), \omega_r(k-1), \mathbf{v}_s(k-1)\right) + \\ + c_v\, \mathbf{v}_s(k) \quad (1)$$

where this equation represents the value of the stator current at instant $(k+1)$ as a non-linear function $f(.)$ of present and previous sampled values of the stator currents, rotor speed and stator voltages. In the second case, described in Fig. 2, the current loop is based on an MRAC. The MRAC uses a reference model that defines the plant output trajectory. The error between the plant output and the reference model is used to identify the machine model. The model of the machine in the two-axis stationary (α, β) coordinates is well known and is given by

$$\dot{\mathbf{i}}_s = \frac{\mathbf{v}_s}{L_{eq}} - \left(\frac{R_s}{L_{eq}} + \frac{M^2\, R_r}{L_{eq}\, L_r^2}\right) \mathbf{i}_s + \left(\frac{M\, R_r}{L_{eq}\, L_r^2} - j\,\frac{\omega_r\, M}{L_{eq}\, L_r}\right) \boldsymbol{\lambda}_r^s \quad (2)$$

$$\dot{\boldsymbol{\lambda}}_r^s = \frac{M\, R_r}{L_r}\, \mathbf{i}_s - \left(\frac{R_r}{L_r} - j\omega_r\right) \boldsymbol{\lambda}_r^s \quad (3)$$

Since the stator current is to be controlled, the following first order linear and stable model is proposed as reference.

$$\dot{i}_{s\alpha\_m} + A_\alpha\, i_{s\alpha\_m} = B_\alpha\, i_{s\alpha\_ref} \quad (4)$$

$$\dot{i}_{s\beta\_m} + A_\beta\, i_{s\beta\_m} = B_\beta\, i_{s\beta\_ref} \quad (5)$$

$i_{s\alpha\_m}$ and $i_{s\beta\_m}$ are the outputs of the reference model, the constants A_α, A_β, B_α and B_β are positive and are selected such that the model response is similar to the desired plant response. After some algebraic manipulations of (2) and (3) the following relationships for the stator voltage are obtained

$$v_{s\alpha} = L_{eq}\, B_\alpha\, i_{s\alpha\_ref} + \left(\left(R_s + \tfrac{M^2\, R_r}{L_r^2}\right) - L_{eq}\, A_\alpha\right) i_{s\alpha} - \\ - \tfrac{M\, R_r}{L_r^2}\, \lambda_{r\alpha}^e - \tfrac{M}{L_r}\, \omega_r\, \lambda_{r\beta}^s \quad (6)$$

$$v_{s\beta} = L_{eq}\, B_\beta\, i_{s\beta\_ref} + \left(\left(R_s + \tfrac{M^2\, R_r}{L_r^2}\right) - L_{eq}\, A_\beta\right) i_{s\beta} + \\ + \tfrac{M\, R_r}{L_r^2}\, \omega_r\, \lambda_{r\alpha}^s - \tfrac{M\, R_r}{L_r}\, \lambda_{r\beta}^s \quad (7)$$

Each stator voltage component is identified by an ANN as shown in Fig. 2. The neural networks present in the controller use the Lyapunov based training algorithm presented in [9], and their performance will be compared with the NARMAX based neurocontroller.

III. ANN STRUCTURE AND TRAINING ALGORITHM

The capabilities of neural networks to approximate a non-linear function are exploited to estimate and control the current in the system. For the MRAC based neurocontroller shown in Fig. 2, 2 multilayer perceptrons (ANN-α and ANN-β) with one hidden layer in each, are used. Each ANN has 4 inputs: $i_{s\alpha}$, $i_{s\alpha\_ref}$, $\lambda_{r\alpha}$ and $\omega_r\lambda_{r\beta}$ for ANN-α and $i_{s\beta}$, $i_{s\beta\_ref}$, $\lambda_{r\beta}$ and $\omega_r\lambda_{r\alpha}$ for ANN-β, 4 hidden neurons with sigmoid activation functions, and one linear neuron in the output layer.

A. Neuron Model

The ANN used in the neurocontroller is a standard multilayer perceptron (MLP) or feedforward neural network, and is formed by one or more neurons interconnected in layers, where the neurons in the same layer share the same inputs. The neurons are composed of the following parts: inputs, weighted adder, activation function and output. By using the notation outlined in [12], the output for neuron j in layer k is represented by the following expression:

$$y_j^k = S\left(\sum_{i=0}^{N^k-1} W_{i,j}^k x_i^k\right) \quad (8)$$

where $S(.)$ is the activation function. In this work the activation function for the neurons in the first layer (hidden layer), is a mono-polar sigmoid, also known as logistic function, described by the following expression

$$S(n) = \frac{1}{1+e^{-n}} \quad (9)$$

B. Training Algorithm

The weights in the ANN determine the relation between the inputs and the outputs, and the ANN can therefore approximate the corresponding response of a multivariable function by adjusting the values of the ANN weights; this process is known as training and can be performed off line using a batch of data, or on line by continually measuring the variables and the corresponding output in the system to be approximated. Backpropagation is the most commonly used algorithm to achieve this by minimizing a cost function that depends on the error e_p between the desired value and the ANN output

$$\epsilon = \frac{1}{2}\sum_{k=1}^{M} e_p^2 \quad (10)$$

M is the number of outputs in the neural network. The backpropagation algorithm could converge into a local minimum, thus a different general training strategy for feedforward Neural Networks is proposed in [10], where the Lyapunov function is used to derive conditions that guarantee convergence of the weights. The following Lyapunov function with

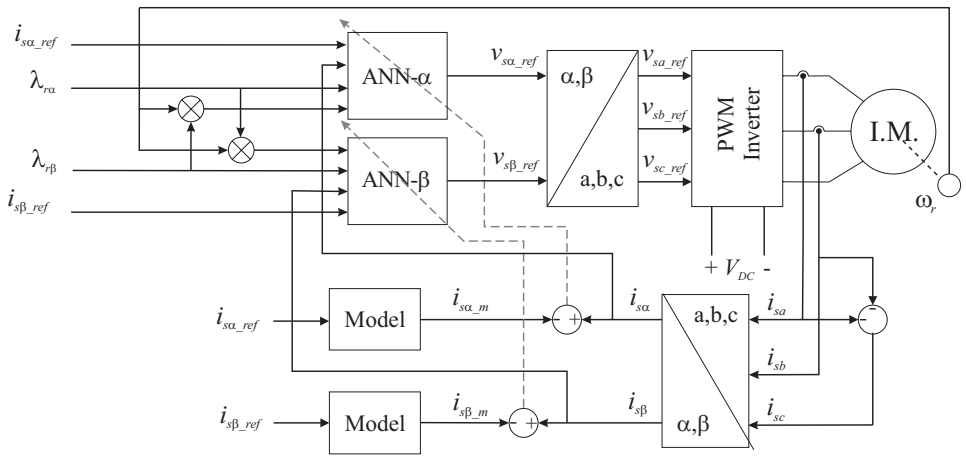

Figure 2: Model for the MRAC based neurocontroller in field oriented coordinates.

respect to the cost function ϵ and to $\partial\epsilon/\partial W$, is employed in defining the training algorithm

$$V = \mu\epsilon + \frac{1}{2}\sigma \left\| \frac{\partial \epsilon}{\partial W_{i,j}^k} \right\|^2 \tag{11}$$

where $\mu > 0$, $\sigma > 0$. Equation (9) is a positive definite function, and if V is minimized then

$$\epsilon = 0 \text{ and } \frac{\partial \epsilon}{\partial W} = 0 \tag{12}$$

This result corresponds with the global minimum in the error surface. The Lyapunov second method or Lyapunov function method requires $\partial V/\partial t$ to be negative definite to assure that the global minimum is asymptotically reached [10][11]. This can be accomplished if an adequate weight adaptation law is chosen as shown in [6], and the resulting general matrix update for the weights is:

$$\dot{W}_{i,j}^k = -\frac{1}{\left(\mu\ I + \sigma\ \frac{\partial^2 \epsilon}{\partial W_{i,j}^k \partial W_{i,j}^{kT}}\right)} \left(\frac{\frac{\partial \epsilon}{\partial W_{i,j}^{kT}}}{\left\| \frac{\partial \epsilon}{\partial W_{i,j}^k} \right\|^2} \right) \left(\zeta \left\| \frac{\partial \epsilon}{\partial W_{i,j}^k} \right\|^2 + \eta \left\| \epsilon \right\|^2 \right) \tag{13}$$

where $\zeta > 0$, $\eta > 0$ control the convergence speed of the training algorithm, and the index k stands for the layer where the weights belong: $k = 1$ for the hidden layer and $k = 2$ for the output layer. For the hidden layer, ΔW is a (4x4) matrix containing the update values for the weights between the inputs and the hidden layer, and for the output layer, ΔW is a (4x1) matrix for the update in the weights between the hidden layer and the output layer. The matrices for the first order and second order derivatives of J are given by:

$$\frac{\partial \epsilon}{\partial W^1} = \begin{bmatrix} \frac{\partial \epsilon}{\partial W_{0,0}^1} & \cdots & \frac{\partial \epsilon}{\partial W_{0,N_H-1}^1} \\ \frac{\partial \epsilon}{\partial W_{1,0}^1} & \ddots & \vdots \\ \vdots & & \vdots \\ \frac{\partial \epsilon}{\partial W_{N_I-1,0}^1} & \cdots & \frac{\partial \epsilon}{\partial W_{N_I-1,N_H-1}^1} \end{bmatrix} \tag{14}$$

$$\frac{\partial \epsilon}{\partial W^2} = \begin{bmatrix} \frac{\partial \epsilon}{\partial W_0^2} \\ \frac{\partial \epsilon}{\partial W_1^2} \\ \vdots \\ \frac{\partial \epsilon}{\partial W_{N_H-1}^2} \end{bmatrix} \tag{15}$$

$$\frac{\partial^2 \epsilon}{\partial W^{1^2}} = \begin{bmatrix} \frac{\partial^2 \epsilon}{\partial W^{1^2}_{0,0}} & \cdots & \frac{\partial^2 \epsilon}{\partial W^{1^2}_{0,N_H-1}} \\ \frac{\partial^2 \epsilon}{\partial W^{1^2}_{1,0}} & \ddots & \vdots \\ \vdots & & \vdots \\ \frac{\partial^2 \epsilon}{\partial W^{1^2}_{N_I-1,0}} & \cdots & \frac{\partial^2 \epsilon}{\partial W^{1^2}_{N_I-1,N_H-1}} \end{bmatrix} \tag{16}$$

$$\frac{\partial^2 \epsilon}{\partial W^{2^2}} = \begin{bmatrix} \frac{\partial^2 \epsilon}{\partial W_0^{2^2}} \\ \frac{\partial^2 \epsilon}{\partial W_1^{2^2}} \\ \vdots \\ \frac{\partial^2 \epsilon}{\partial W_{N_H-1}^{2^2}} \end{bmatrix} \tag{17}$$

where ∂ is the second partial derivative of the cost function ϵ with respect to the weight $j-th$ in the output layer.

Each element in the following expression is obtained as the dot product of each row vector in (7) and its transpose.

$$\left\| \frac{\partial \epsilon}{\partial W^1} \right\|^2 = \begin{bmatrix} \sum_{j=0}^{N_H-1} \left(\frac{\partial \epsilon}{\partial W_{0,j}^1} \right)^2 \\ \sum_{j=0}^{N_H-1} \left(\frac{\partial \epsilon}{\partial W_{1,j}^1} \right)^2 \\ \vdots \\ \sum_{j=0}^{N_H-1} \left(\frac{\partial \epsilon}{\partial W_{N_I-1,j}^1} \right)^2 \end{bmatrix} \tag{18}$$

The elements in (12) are obtained in a similar manner from the column vectors in (8).

$$\left\| \frac{\partial \epsilon}{\partial W^2} \right\|^2 = \sum_{j=0}^{N_H-1} \left(\frac{\partial \epsilon}{\partial W_j^2} \right)^2 \tag{19}$$

The generic weight $W_{i,j}$ is updated according to the following equation.

$$\Delta W_{i,j}^1 = \frac{x_i W_j^2 \left(W_j^2 \alpha_j^2 + e\beta_j \right)}{\mu + \sigma \left[x_i W_j^2 \left(W_j^2 \alpha_j^2 + e\beta_j \right) \right]} \cdot \cdot \frac{\zeta \sum_{j=0}^{N_H-1} \left(\frac{\partial \epsilon}{\partial W_{0,j}^1} \right)^2 + \eta \left(\frac{1}{2} e^2 \right)^2}{\sum_{j=0}^{N_H-1} \left(\frac{\partial \epsilon}{\partial W_{0,j}^1} \right)^2} \tag{20}$$

For the activation function (9) the expressions for α_j, and β_j in (20) are:

$$\alpha_j = S_j(1 - S_j) \tag{21}$$

$$\beta_j = 2(S_j^3 - 3S_j^2 + S_j) \tag{22}$$

IV. Simulations

The proposed training algorithm was first tested by simulation using a real time program execution, to verify its suitability for its practical application in the control of induction machines.

A. Conditions for the simulations

To attain real time execution the simulation of the control algorithms were programmed in C language, the induction machine was simulated by solving the differential equations that model the machine, using a fixed time step fourth order Runge-Kutta Ordinary Differential Equations (ODE) integrator. The simulations were executed on a Digital Signal Processor ADSP-21369 running at 333 MHz and was programmed using the manufacturer Integrated Development Environment (IDE) for that device, VisualDSP++ 5.1. The induction machine was simulated in the (α, β) reference frame and the voltage commands to the PWM model were in natural (a, b, c) coordinates, as shown in Fig. 1. The parameters of the simulated machine are shown in Table I in the Appendix. As mentioned in Section II, the structure of the ANN used for the NARMAX modeling of the induction machine is similar to the one presented in [5], having 8 inputs, 8 neurons with logistic activation functions in the hidden layer, and 2 neurons with linear activation functions in the output layer. For the MRAC based neurocontroller, during each control cycle the inputs are fed forward through the ANN to provide the demanded voltage to the PWM stage. The power stage was simulated for a three-phase PWM and a 200 V DC bus link, driven by the ANN current loop controller. The simulated PWM switching and sampling frequency were 10 kHz, and the command to the PWM was fed as a percentage of the maximum duty cycle. The training parameters for the Lyapunov algorithm used in the simulation and in the experimental tests for both control schemes were heuristically selected to be $\eta = 0.01$, $\zeta = 1.0$, $\mu = 10.0$ and $\sigma = 0.01$.

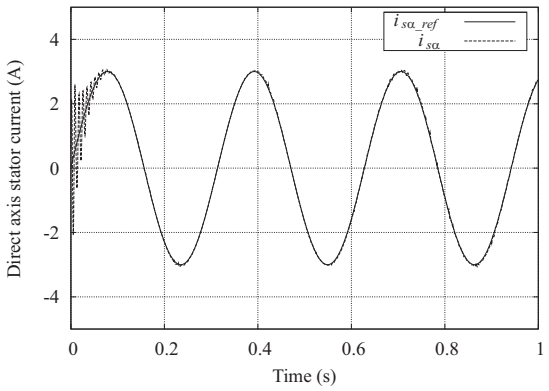

Figure 3: Simulated direct axis stator current and reference current using the NARMAX approach for initial ANN weights with random values.

B. Results of the simulation

The timing for the current control loop algorithm is similar to the one presented in [5], the control cycle is started by an acquisition request. After the DSP has retrieved all the measured control variables, the ANN algorithm is executed. First the forward propagation of the inputs into the ANN is executed, after that, the training of the ANN is executed. For the present implementation of the algorithm the ANN training is performed every other control cycle, to avoid over training of the neural network. At the end of the control cycle the simulated PWM module is fed with the new set of duty cycles in natural coordinates, but they take effect at the beginning of the next control cycle, which also corresponds to the next PWM switching cycle. The first test was done for the NARMAX based neurocontroller using 20 rad/s sinusoidal references for phases a, b and c. Figure 3 shows the direct axis stator current component and its corresponding reference, when the weights matrices were initialized using the compiler pseudo random number generator, with values in the range $[-0.1; 0.1]$. Additionally, there is no pre-training or commissioning stage prior to the test. At the beginning of the test there was a long transient present in the stator currents, but after 100 ms the ANN had succeeded in tracking the stator current reference. Figure 4 shows the response when the weights matrices are loaded with values obtained in a previous run of the algorithm and the training is stopped after 1 s. The error between the simulated stator current and the reference signal is shown in Fig. 5.

Figures 6 to 8 show the results of applying a similar test to the MRAC based neurocontroller. In this case the parameters for the reference model are described by (4) and (5) were $\alpha = \beta = 5000$. Again the weights matrices were initialized with pseudo random numbers in the range [-0.1;0.1]. The oscillations present in the stator current, shown in Fig. 6, are smaller than in the previous case, shown in Fig. 3, but they last longer due to the dynamics of the first order reference model. In the second test the weights matrices were initially loaded

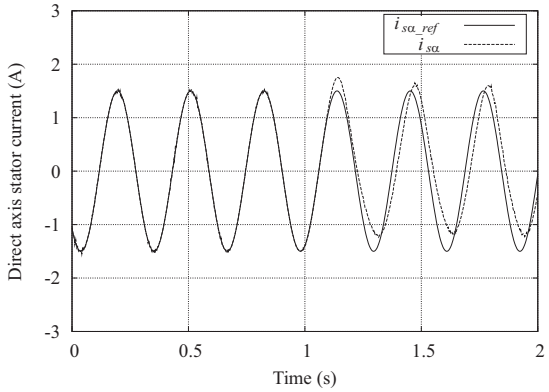

Figure 4: Simulated direct axis stator current and reference current using the NARMAX approach when the training is stopped at 1 s.

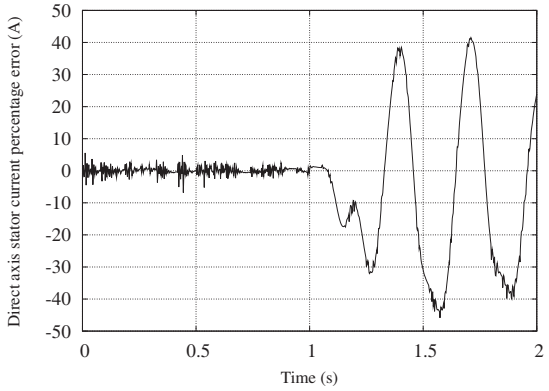

Figure 5: Percentage error between the reference and simulated direct axis stator current shown in Fig. 4.

with values from a previous run and after 1 s the training was stopped. The error between the simulated stator current and the reference value is shown in Fig. 8.

V. Experimental results

A view of the experimental setup is shown in Fig. 9. The power stage and control boards were developed by the authors as reported in [13], [14]. The control board is developed around the same ADSP-21369 used for the simulations, whose operation is complemented with an FPGA Spartan3 from Xilinx which is in charge of: mapping the PWM ports and generate the PWM signals, converting the serial digital data sent from the sensors boards to parallel format and measuring the motor speed through an optical encoder attached to the machine shaft. The inverter uses six 100 A, 1200 V IGBTs as switching devices and feeds power to a 220 V 1 1/2 HP induction machine; the machine name-plate is outlined in the Appendix.

A. NARMAX based neurocontroller

Figure 10 shows the stator current when the ANN weights are initialized with pseudo random numbers in the range $[-0.1; 0.1]$. The behavior in the simulation, Fig. 3, resembles the results obtained in the experimental test, where there

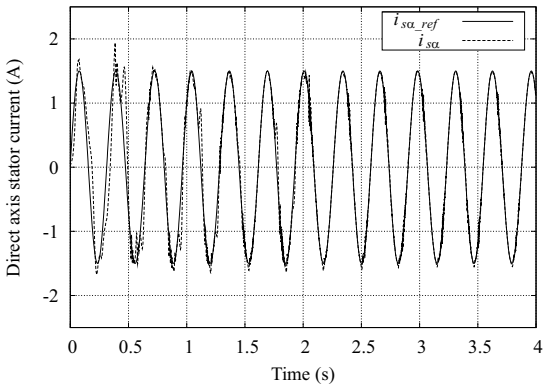

Figure 6: Simulated direct axis stator current and reference current using the MRAC approach for initial ANNs weights with random values.

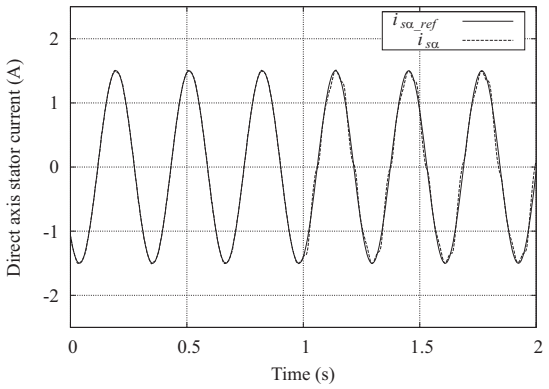

Figure 7: Simulated direct axis stator current and reference current using the MRAC approach when the training is stopped at 1 s.

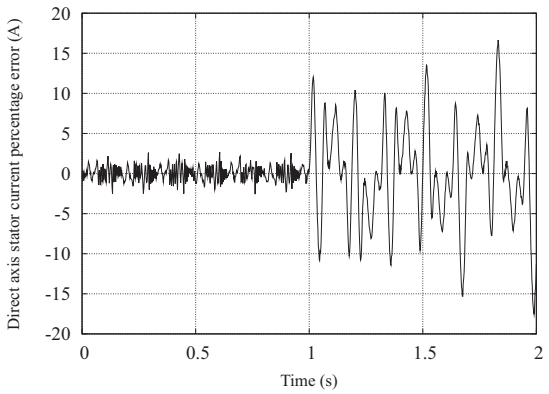

Figure 8: Percentage error between the reference and simulated direct axis stator current shown in Fig. 7.

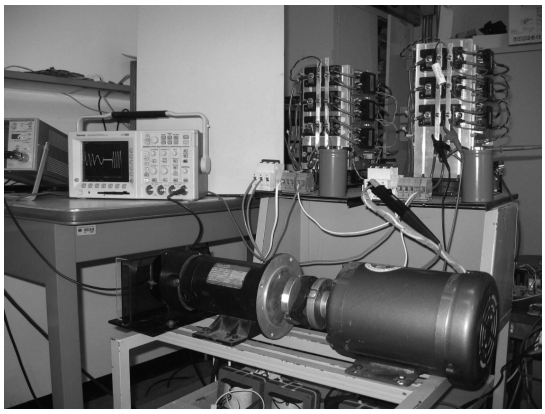

Figure 9: View of the experimental test setup for the Lyapunov based ANN training.

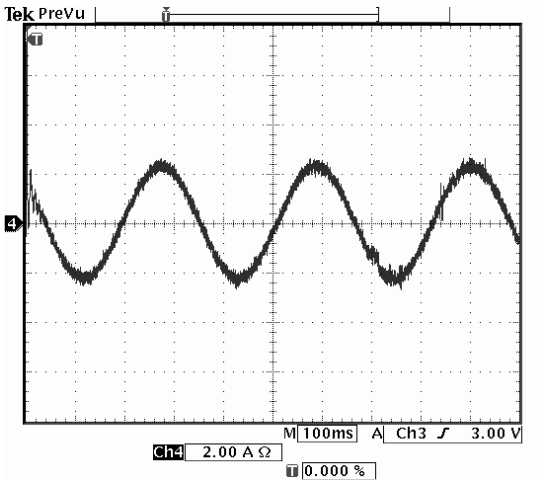

Figure 10: Experimental phase a stator current when using the NARMAX approach for initial ANN weights with random values.

is an initial transient in the measured stator current and quickly follows the demanded value. Instead of evaluating the evolution of each ANN weight separately, the Frobenius norm can be used as a metric of the "size" of matrix of weights (W^1 and W^2) [15]. Figure 11 shows the norms evolution for a 1800 s execution of the algorithm with the same reference shown in Fig. 10. After 1800 s the Frobenius norm for the weights matrix in the hidden layer ($N_F(W)$) is still changing, i.e. the learning has not been stabilized even after 1800 s, even thought the neurocontroller keeps tracking the reference signal. This suggests that a global minimum for the error function still has not been reached.

B. MRAC based neurocontroller

Figure 12 shows the current in phase a of the induction machine when the weights matrices for ANN-α and for ANN-β were randomly initialized. Figure 13 shows the response of the neurocontroller when the ANNs are loaded with pretrained weights and the training is stopped after 1 s. This result closely resembles that obtained in the simulation. The percentage error between the reference and the actual signal is depicted in Fig.

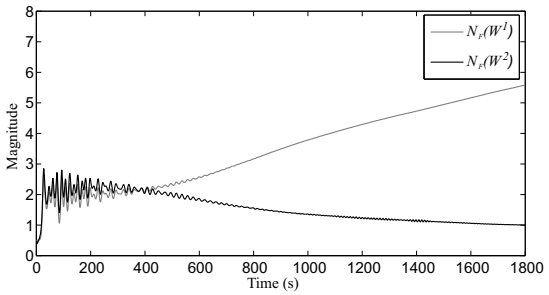

Figure 11: Experimental evolution of the Frobenius norm of weights matrices W^1 and W^2.

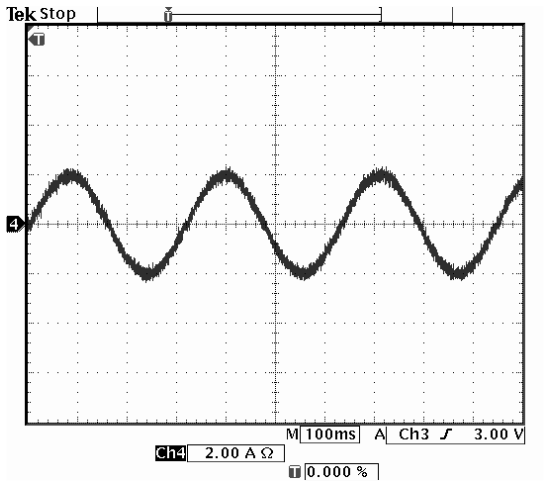

Figure 12: Experimental phase a stator current when using the MRAC approach for initial ANN weights with random values.

14 and is in the same order of magnitude observed in Fig. 8 for the simulation. Figure 15 shows the evolution of the Frobenius norm for the weights matrices in the hidden layer and in the output layer of ANN-α and ANN-β. In contrast with the result obtained for the NARMAX based neurocontroller, the matrix norms tend to stabilize indicating that both neural networks have reached a minimum in the error surface defined by the cost function (10). A comparison between expressions (1) and (6) and (7) shows that in the MRAC scheme the functions to be identified by the neural networks are simpler than in the NARMAX case, resulting in less time to obtain a stable final value for the Frobenius norms of weights matrices as seen in Fig. 15.

VI. CONCLUSION

In this work the use of a Lyapunov based training algorithm has been presented for a MRAC controller based in the state variables feedback (see Fig. 2). The proposed controller was compared against another one based in the NARMAX modeling description of the induction machine (see Fig. 1), described in [5]. The performance of both controllers was verified by numerical simulations in C language and later corroborated using an experimental test system. The proposed MRAC controller achieves a faster reduction of the current tracking error than the obtained with the NARMAX based controller. Also when training is stopped MRAC based current

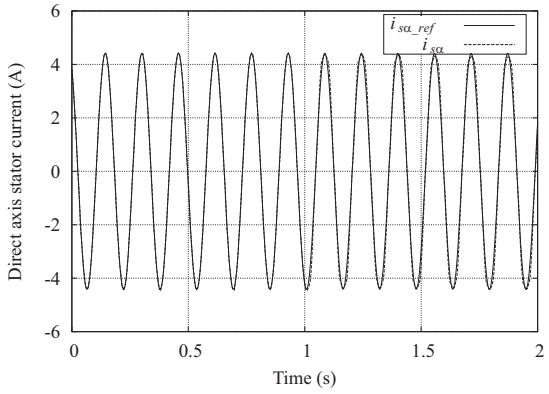

Figure 13: Experimental direct axis stator current and reference current using the MRAC approach when the training is stopped at 1 s.

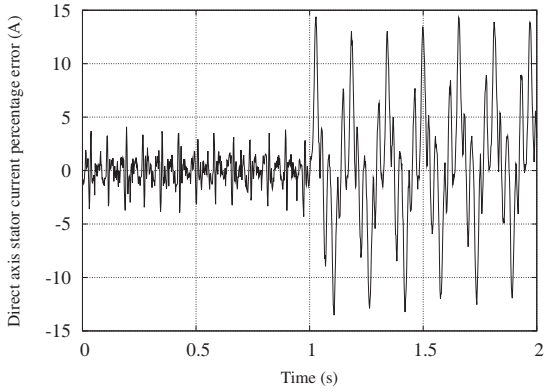

Figure 14: Percentage error between the reference and actual direct axis stator current shown in Fig. 13.

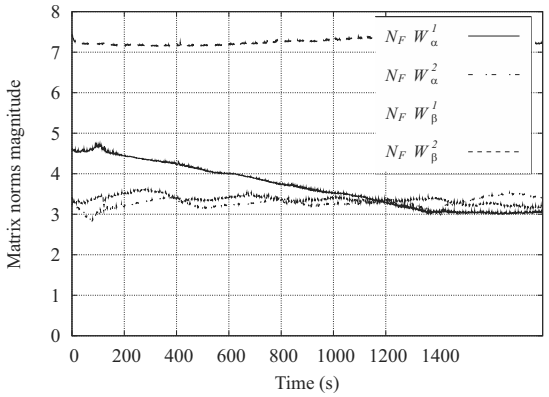

Figure 15: Experimental evolution of the Frobenius norm of weights matrices W^1 and W^2 in the MRAC case.

Table I: Induction machine parameters used in the simulation

PARAMETER	VALUE
R_s	1.04 Ω
R_r	1.3 Ω
M	0.662 H
$L_s = L_r$	0.6753 H
J	0.0027 kg m^2

controller exhibits a smaller tracking error showing that a best identification of the machine parameters was obtained. The implementation of the Lyapunov based training algorithm guarantees the stability of the MRAC controller as can be appreciated from the Frobenius norm evolution after a long test of 1800 s, which involves 18.10^6 weights adaptations. When is applied to the NARMAX controller, however, the same training algorithm does not guarantee the uniqueness of the solution and numerical instability may arise in a practical implementation since, as shown in Fig. 11, the wandering of the weights may lead to neurons output saturation. The simpler structure of the MRAC based controller which uses 2 ANNs with just 4 inputs, combined with the Lyapunov training algorithm give the best results with a faster adaptation, even when the ANN is initialized with randomly generated weights, and long term stability.

VII. Appendix

The parameters for the induction machine used in the simulation are shown in Table I. The specs for the induction machine in the test rig are: CAT No M3550 (BALDOR), Specs: 35A11-82, 1 $\frac{1}{2}$ HP, 208-230/460 Volts. 5-4.6/2.3 Amps, 60 Hz, 3450 rpm.

Acknowledgment

The authors gratefully acknowledge the support of Prometeo Project, Secretaría de Educación Superior, Ciencia, Tecnología e Innovación, and Universidad Politécnica Salesiana, both in República del Ecuador, as well as Decanato de Investigación y Desarrollo de la Universidad Simón Bolívar and FONACIT Research Project ♯ 2011000970 both in Venezuela.

References

[1] D. Holmes, B. McGrath, and S. Parker, "Current regulation strategies for vector-controlled induction motor drives," *Industrial Electronics, IEEE Transactions on*, vol. 59, pp. 3680–3689, Oct 2012.

[2] N. Kutasi, A. Kelemen, and M. Imecs, "Vector control of induction motor drives with model based predictive current controller," in *Computational Cybernetics, 2008. ICCC 2008. IEEE International Conference on*, pp. 21–26, Nov 2008.

[3] A. Ravi Teja, C. Chakraborty, S. Maiti, and Y. Hori, "A new model reference adaptive controller for four quadrant vector controlled induction motor drives," *Industrial Electronics, IEEE Transactions on*, vol. 59, pp. 3757–3767, Oct 2012.

[4] P. Alsina and N. Gehlot, "Neuro-adaptive control of induction motor stator current," in *Industrial Electronics, Control, and Instrumentation, 1995., Proceedings of the 1995 IEEE IECON 21st International Conference on*, vol. 2, pp. 1434–1439 vol.2, Nov 1995.

[5] J. Restrepo, J. Viola, R. Harley, and T. Habetler, "Induction machine current loop neuro controller employing a Lyapunov based training algorithm," in *Power Engineering Society General Meeting, 2007. IEEE*, pp. 1–8, June 2007.

[6] S. Gadoue, D. Giaouris, and J. Finch, "An experimental assessment of a stator current MRAS based on neural networks for sensorless control of induction machines," in *Sensorless Control for Electrical Drives (SLED), 2011 Symposium on*, pp. 102–106, Sept 2011.

[7] X. Dai and X. Wang, "Neural network inverse control of current-fed induction motor," in *Industrial Electronics, 2008. ISIE 2008. IEEE International Symposium on*, pp. 431–436, June 2008.

[8] C. Macnab, "Getting weights to behave themselves: achieving stability and performance in neural-adaptive control when inputs oscillate," in *American Control Conference, 2005. Proceedings of the 2005*, pp. 3192–3197 vol. 5, June 2005.

[9] X. Yu, M. Efe, and O. Kaynak, "A general backpropagation algorithm for feedforward neural networks learning," *Neural Networks, IEEE Transactions on*, vol. 13, pp. 251–254, Jan 2002.

[10] J. C. Viola and J. A. Restrepo, "Lyapunov-based training algorithm applied to a continually on line-trained ANN used in the current-loop control of a single-phase switched rectifier," *International Journal of Adaptive Control and Signal Processing*, vol. 22, pp. 609–625, Aug. 2008.

[11] J. Restrepo, B. Burton, R. Harley, and T. Habetler, "ANN based current control of a VSI fed AC machine using line coordinates," in *Devices, Circuits and Systems, 2004. Proceedings of the Fifth IEEE International Caracas Conference on*, vol. 1, pp. 225–229, Nov 2004.

[12] M. Gupta, L. Jin, and N. Homma, *Static and Dynamic Neural Networks: From Fundamentals to Advanced Theory*. John Wiley & Sons, 2003.

[13] M. Gimenez, V. Guzman, J. A. Restrepo, J. Aller, A. Bueno, J. C. Viola, A. Millan, and A. Cabello, "PLATAFORMA: Development of an integrated dynamic test system to determine power electronics systems performance," *Revista de la Facultad de Ingenieria - UCV*, vol. 23, no. 3, pp. 91 – 102, 2008.

[14] J. Viola, J. Restrepo, F. Quizhpi, M. Gimenez, J. Aller, V. Guzman, and A. Bueno, "A flexible hardware platform for applications in power electronics research and education," in *Electrical Power Energy Conference (EPEC), 2014 IEEE*, pp. 1–6, Nov. 2014.

[15] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*. Philadelphia: Soc. Industrial and Appl. Math., 1996.