CBRN 2005 - VII Brazilian Congress of Neural Networks
VIII Neural Networks School
Natal, Rio Grande do Norte, Brazil, October 16–19, 2005

Induction Machine Speed Neural Estimator Implemented in Programmable Architecture

L.F. Mouzinho[1], J.V. FonsecaNeto[2], B.A. Luciano[3] and R.C.S. Freire[3]

[1]Department of Electronic and Electrical Engineering,
Maranhão´s Federal Center of Technologic Education,
E–mail: mouzinho@dee.cefet-ma.br.

[2]Department of Electrical Engineering
Federal University of Maranhão,
E–mail: jviana@dee.ufma.br.

[3]Department of Electrical Engineering
Federal University of Campina Grande,
E–mail: benedito@dee.ufcg.edu.br, rcsfreire@dee.ufcg.edu.br.

Abstract – *This work presents a state observer based on Artificial Neural Networks (*ANN's*) and a Programmable Architecture to estimate an Induction Machine (*IM*) speed . Indirect measurement system is the observer's physical accomplishment formed by hardware components, voltage/current meters and digital circuit, by the software and the estimation algorithm. Speed obtained by simulation based on mathematical models and by estimation algorithm implementation in PSoC programmable circuit is compared to machine's speed measured values by a tachometer.*

Keywords – *artificial neural network, estimation, indirect measurement, induction machine speed, programmable architecture, PSoC.*

I. INTRODUCTION

Electromechanical sensors replacement for indirect measurements systems is an alternative to improve induction motor performance. In our case, the tachometer is replaced by a state estimator that computes Induction Machine, *IM*, rotor speed through stator voltages and currents. This action can be characterized as an indirect measurement signal processing because the desired or estimated measurement is a function of direct measurements. In general terms, the proposed indirect measurement system is assembled based on a hardware and software components set. Set elements are voltage and current meters and an programmable architecture that is in charge for managing estimation algorithm trough voltage and current signal processing. Generally, in this text, observer term is used more often in design and indirect measurement systems in the implementation steps of the speed estimation system.

State observer models developed to estimate an induction machine speed have as input the voltage and current measured values [1]. Starting from these measured values, non electric variables such as speed, angular displacement and induction machine torque are considered. Replacement of traditional speed measurement mechanisms for electronic devices which perform indirect measurements, contributes to increase induction motor´s operational robustness, as to the tachometer concern, and to reduce implementation costs in control project and in non electric variables supervision.

An indirect measurement system to obtain *IM* speed values is implemented on a Programmable System on Chip (PSoC). A non-conventional observer based on Neural Networks estimates speed for a machine's wide operational range and its results are compared to measurements which came from tachometer. Following the guidelines practical implementation aspects are considered and discussed [2].

Besides general training abilities, by increasing the capacity to prepare the observer when working with adverse operation situations and faults tolerance, due to the high relationship degree between the involved variables in estimation process, specific qualities which justify Neural Networks application in *IM* are emphasized. Most of real systems present some non linearity and those systems linear modelling doesn't represent system's total dynamics, and limitations of linear models also limit accuracy range of indirect measurements. In this work, one of the considerations used in the estimation is an unknown system´s existence which is linear or whose behavior can be linearized within certain operation area. Artificial Neural Networks (*ANN's*) have a quite promising use in the identification of non linear dynamical systems. *ANN's* come to be a proper tool for non-linear systems modelling due to their learning ability [3], [4] and [5].

II. PLANT AND ARTIFICIAL NEURAL NETWORK MODELS

Plant and Artificial Neural Network models represent *IM* and state estimator, respectively. These models main charac-

teristics are discussed in subsections A and B.

A. Induction Machine Model

Plant model inputs are voltage and current in a stationary *dq0* reference system and the output is rotor flux. Model equations in terms of the vectorial quantities are specified in [6]-[9]. Model main equations are written as induced voltages:

$$\dot{\vec{\lambda}}_d = \frac{L_r}{L_m}(\vec{v_s} - R_s\vec{i_s} - \sigma L_s\dot{\vec{i}_s}) \quad (1)$$

and

$$\dot{\vec{\lambda}}_a = (-\frac{1}{T_r} + \omega_r j)\vec{\lambda_a} + \frac{L_m}{T_r}\vec{i_s} \quad (2)$$

B. Artificial Neural Networks Model

ANN's are the indirect measurement system´s software core and they are trained to obtain the rotor flux estimation. Estimated flux error and model flux are used for tuning speed estimator weights.

Training procedure represents the system´s direct dynamics is show in Fig. 1. Neural model is coupled in parallel with the plant and prediction error is used as a signal to tune neural network weights. In this structure, the plant directly supplies its output values to be used in neural estimator adaptive learning. If *ANN's* are a multilayer perceptron type, prediction error is the start point to derive the training algorithm[4].

ANN models plant dynamics and its input is a sequence of delayed time signals. Also, it is possible to use representation through dynamic neurons, changing the model´s input/output description into steady state approach[10]. A state observer´s general view to speed estimation of an *IM* is shown in Figs. 1 and 2.

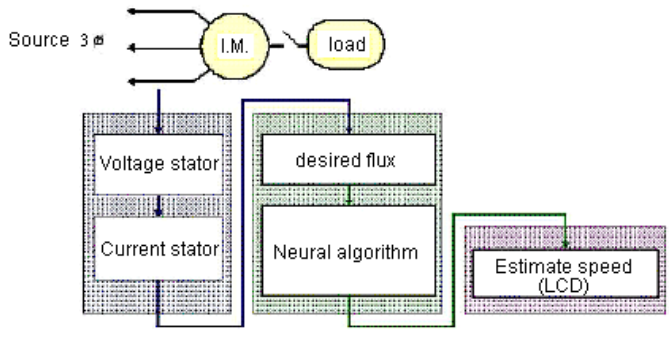

Fig. 1. Structure of artificial neural network system for speed estimation

III. NEURAL ESTIMATOR

Amongst neuronal identification forms, direct modeling was used to represent the neural network structure which estimates the rotor speed, ω_r. In the Fig. 2, there is the induction

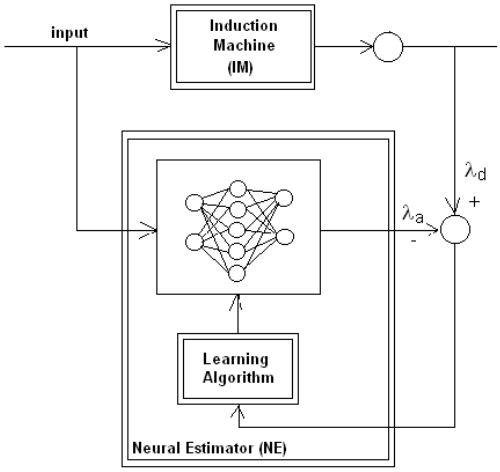

Fig. 2. Model of artificial neural networks estimation

machine block , whose input is represented by rotor voltages and current. In this block output λ_d is obtained by Equation (1) and representing the desired flux. In neural network block output λ_a is obtained by Equation (2), representing actual flux. Induction machine and neural network blocks are in parallel[5], .

When actual flux λ_a approaches to the desired flux λ_d, neural network starts to supply the signals delayed in time its proper input. The error between desired and actual (estimated) fluxes is used by the learning algorithm to adjust one of the neural network weights, in this case, ω_r. When motor parameters are known, neural and induction machine models must coincide. However, any difference between the used speed in neural model and IM speed can automatically result in error between the outputs of the two estimators. This error between desired and actual fluxes is used to brings neural networks´s model weights up to date, or either rotor speed, ω_r, in Equation (2). This approach is shown in Figure 2, having been the backpropagation algorithm derived from such a form that the estimator of Equation (2) follows the next possible the estimator to Equation (1). To obtain the algorithm backpropagation, the data standards model of the Equation (2) first is derived using the backward difference method [11] and [4].

The rotor flux (λ_r) instantaneous variation range in T relation to the instant in which $T = k$, it's given by $\lim_{\Delta T\to 0}\frac{\Delta\lambda_r}{\Delta T} = \lim_{\Delta T\to 0}(\frac{\vec{\lambda}_r(k)-\vec{\lambda}_r(k-1)}{T})$ then,

$$\dot{\vec{\lambda}}_a(k) = \frac{\vec{\lambda}_a(k) - \vec{\lambda}_a(k-1)}{T} \quad (3)$$

Applying the recursive method in the right side of the Equation (3), equaling the Equation (2), it follows,

$$\frac{\vec{\lambda}_a(k)-\vec{\lambda}_a(k-1)}{T}=(\frac{-1}{T_r}I+\omega_r J)\vec{\lambda}_a(k-1)+\frac{L_m}{T_r}\vec{i}_s(k-1) \tag{4}$$

then,

$$\vec{\lambda}_a(k)=I\vec{\lambda}_a(k-1)+(\frac{-1}{T_r}I+\omega_r J)T\vec{\lambda}_a(k-1)+\frac{L_m}{T_r}T\vec{i}_s(k-1) \tag{5}$$

Organizing terms in relation to matrices I, J and variables $\vec{\lambda_a}$ and $\vec{i_s}$,

$$\vec{\lambda}_a(k)=(1-\frac{T}{T_r})I\vec{\lambda}_a(k-1)+\omega_r TJ\vec{\lambda}_a(k-1)+\frac{L_m}{T_r}T\vec{i}_s(k-1) \tag{6}$$

Equation (6) can be written in the following form:

$$\vec{\lambda}_a=W_1X_1+W_2X_2+W_3X_3 \tag{7}$$

or

$$\vec{\lambda}_a=\sum_{i=1}^{3}W_iX_i, \tag{8}$$

being $W_1=1-\frac{T}{T_r}$, $X_1=I\vec{\lambda}_a(k-1)$, $W_2=\omega_r T$, $X_2=J\vec{\lambda}_a(k-1)$, $W_3=\frac{L_m}{T_rT}$, $X_3=I\vec{i}_s(k-1)$, $I=\begin{bmatrix}1&0\\0&1\end{bmatrix}$, $J=\begin{bmatrix}0&-1\\1&0\end{bmatrix}$ and T is the sampling period.

Equation (7) can be represented as a neural model, with two layers, being W_1, W_2 and W_3 the weights of the neural network; X_1, X_2 and X_3, inputs and $\vec{\lambda}_a$, output, Figure 3, [6].

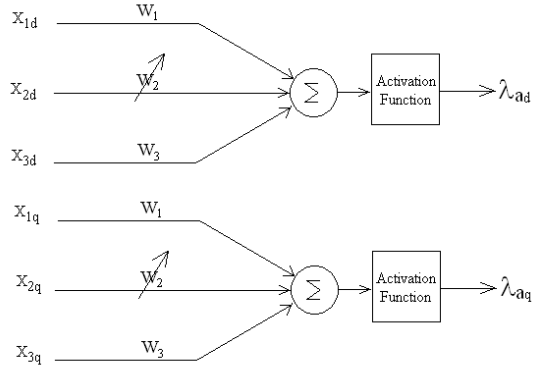

Fig. 3. Neural network

The output error between desired flux and actual flux, mathematically, is given by:

$$\vec{\varepsilon}(k)=\vec{\lambda}_d(k)-\vec{\lambda}_a(k) \tag{9}$$

Neural network weights W_1 and W_3 are considered constant, therefore depend on machine parameters and W_2 depends on the machine speed [7].

Synaptic weights ($W1$, $W2$, $W3$) are adjusted to minimize the energy function, [7]-[10]. Instantaneous value of the error energy for these neurons is defined as [7],

$$E=\frac{1}{2}\vec{\varepsilon^2}(k), \tag{10}$$

or,

$$E=\frac{1}{2}\varepsilon^{'}\varepsilon=\frac{1}{2}\|\vec{\lambda}_d(k)-\vec{\lambda}_a(k)\|^2 \tag{11}$$

The correction $\Delta W_2(k)$ applied to backpropagation algorithm for synaptic weight W_2 is determined, [10]:

$$\Delta W_2(k)\propto-\frac{\partial E}{\partial W_2} \tag{12}$$

Through chain rule, gradient $\frac{\partial E}{\partial W_2}$ can be express as, [10]:

$$\frac{\partial E}{\partial W_2}=\frac{\partial E}{\partial \varepsilon}\frac{\partial \varepsilon}{\partial X_2}\frac{\partial X_2}{\partial \lambda_a}\frac{\partial \lambda_a}{\partial W_2}, \tag{13}$$

being $\frac{\partial E}{\partial W_2}$, the sensibility factor that determines search direction in weights space for synaptic weight W_2, [10].

Differentiating Equation (10) both sides in relation the ε , is

$$\frac{\partial E}{\partial \varepsilon}=\vec{\varepsilon} \tag{14}$$

Substituting $\vec{\lambda}_a$ from Equation $X_2=J\vec{\lambda}_a(k-1)$ in Equation (9) and differentiating this equation's both sides in relation to X_2, is

$$\frac{\partial \varepsilon}{X_2}=-1 \tag{15}$$

If $X_2=J\vec{\lambda}_a(k-1)$, then

$$X_2=f(\vec{\lambda}_a) \tag{16}$$

Differentiating Equation (16) in relation to $\vec{\lambda}_a$, there is

$$\frac{\partial X_2}{\partial \vec{\lambda}_a}=f'(\vec{\lambda}_a) \tag{17}$$

Differentiating Equation (7) in relation W_2, is produced

$$\frac{\partial \lambda_a}{\partial W_2}=X_2 \tag{18}$$

Substituting Equations (14), (15) and (17) in Equation (13),

$$\frac{\partial E}{\partial W_2}=\varepsilon(-1)f'(\vec{\lambda}_a)X_2 \tag{19}$$

Correction $\Delta W_2(k)$ applied to W_2, defined as delta rule, is given by:

$$\Delta W_2(k) = -\eta \frac{\partial E}{\partial \vec{\lambda}_a}, \tag{20}$$

being η backpropagation algorithm's learning range and the negative signal indicating gradient descending in weights space to find a direction for weight change in order to reduce error value, ε[10].

Substituting Equation (19) in (20), is

$$\Delta W_2(k) = -\eta \varepsilon (-1) f'(\vec{\lambda}_a) X_2, \tag{21}$$

$$\Delta W_2(k) = -\eta \delta(k) X_2, \tag{22}$$

being $\delta(k)$ the local gradient data for

$$\delta(k) = -\frac{\partial E}{\partial \vec{\lambda}_a} \tag{23}$$

Differentiating Equation (11) in relation to $\vec{\lambda}_a$,

$$\frac{\partial E}{\partial \vec{\lambda}_a} = \frac{1}{2} \frac{\partial [(\vec{\lambda}_d(k) - \vec{\lambda}_a(k))' (\vec{\lambda}_d(k) - \vec{\lambda}_a(k))]}{\partial \vec{\lambda}_a(k)} \tag{24}$$

$$\frac{\partial E}{\partial \vec{\lambda}_a} = \frac{1}{2} \frac{\partial [B + C]}{\partial \vec{\lambda}_a(k)}, \tag{25}$$

being $B = \vec{\lambda}_d(k)' \vec{\lambda}_d(k) - \vec{\lambda}_d(k)' \vec{\lambda}_a(k)$ and $C = -\vec{\lambda}_a(k)' \vec{\lambda}_d(k) + \vec{\lambda}_a(k)' \vec{\lambda}_a(k)$, and $\vec{\lambda}_d(k)' \vec{\lambda}_a(k) = \vec{\lambda}_a(k)' \vec{\lambda}_d(k)$,

$$\frac{\partial E}{\partial \vec{\lambda}_a} = \frac{1}{2} \frac{\partial [(\vec{\lambda}_d(k)^2)' - (2\vec{\lambda}_d(k))' \vec{\lambda}_a(k) + (\vec{\lambda}_a(k)^2)']}{\partial \vec{\lambda}_a(k)}, \tag{26}$$

deriving

$$\frac{\partial E}{\partial \vec{\lambda}_a} = \frac{1}{2}(-2\vec{\lambda}_d(k)' + 2\vec{\lambda}_a(k)') \tag{27}$$

$$\frac{\partial E}{\partial \vec{\lambda}_a} = (-\vec{\lambda}_d(k)' + \vec{\lambda}_a(k)'), \tag{28}$$

substituting Equation (28) in (23),

$$\delta(k) = (\vec{\lambda}_a - \vec{\lambda}_d)' \tag{29}$$

Substituting Equation (23) and considering $X_2 = J\vec{\lambda}_a(k-1)$ in Equation (22),

$$\Delta W_2 = -\eta (\vec{\lambda}_a(k) - \vec{\lambda}_d(k))' J \vec{\lambda}_a(k-1) \tag{30}$$

In Figure 4, it is show the new weight,

$$W_2(k) = W_2(k-1) + \eta \Delta W_2(k), \tag{31}$$

being η, the training coefficient and k coefficient is developed from 1 for each sweeping through input-output set[7].

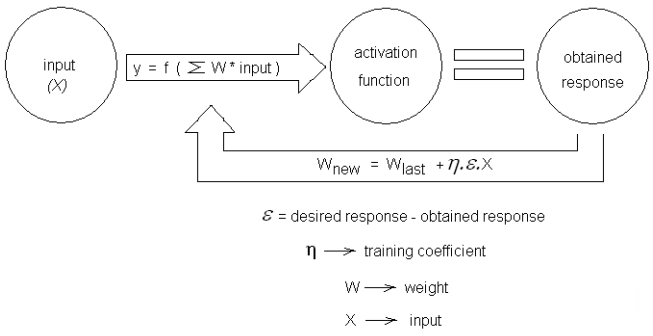

Fig. 4. Diagram for training of perceptron

The delta rule in Equation (22) is modified in to increase the learning range without oscillations, including itself, a moment term, as Equation (32)[10].

$$\Delta W_2(k) = -\eta \delta(k) X_2 + \alpha \Delta W_2(k-1) \tag{32}$$

The coefficient α, moment constant, determines previous weights modifications effect in actual weight. However, it is better to use Equation (32) instead of Equation (31).

Knowing $W_2 = \omega_r T$, then $\omega_r = \frac{W_2}{T}$, varying ω_r,

$$\Delta \omega_r = \frac{\Delta W_2}{T} \tag{33}$$

And, finally, substituting Equation (32) in Equation (33), it produces the rotor estimate speed which is given by:

$$\Delta \omega_r = -\frac{1}{T} \eta \delta(k) X_2 + \frac{1}{T} \alpha \Delta W_2(k-1), \tag{34}$$

$$\hat{\omega}_r(k) - \hat{\omega}_r(k-1) = -\frac{1}{T} \eta \delta(k) X_2 + \frac{1}{T} \alpha \Delta W_2(k-1) \tag{35}$$

then, the estimate speed is, [7]-[12],

$$\hat{\omega}_r(k) = \hat{\omega}_r(k-1) - \frac{1}{T} \eta \delta(k) X_2 + \frac{1}{T} \alpha \Delta W_2(k-1) \tag{36}$$

IV. PROGRAMMABLE ARCHITECTURE

The neural estimator is implemented on a programmable architecture for *IM* speed values indirect measurement. This neural estimator model real time algorithm which is codified in C language of the specified architecture. Estimated speed values via neural network are visualized through a LCD.

V. SIMULATION, EXPERIMENTS AND ANALYSIS

A computational simulation was realized by of the tool *Matlab/Simulink* among the following estimators of rotor speed: estimator via flux; estimator via emf; estimator via flux using *MRAS*; estimator via emf using *MRAS* and estimator via neural networks.

The developed algorithms are based on the equations of the induction machine in the reference frame. The equations of the rotor flux are obtained of the measurements of the stator currents and stator voltage. As result of the analysis, it is verified that all the techniques in analysis present oscillations in the beginning of the time in the waveform. The estimate speeds for the flux and emf present more oscillations until if approximate of the reference value. In the flux using *MRAS* estimate, the estimate speed approached faster of the reference value. The estimate emf using *MRAS* was distant of the reference speed (25% below). In the simulation of the estimator neural, initially were specified several values for the parameters of the network, only for $\alpha = 0.6$, $\eta = 0.5$, it was obtained the estimate speed converging for the reference value. Mouzinho [13] comments that the better estimator speed ω_r it was it estimator of the technique *MRAS* with flux. This estimator converged more quickly for the reference value, after the estimator neural with the reference value a little later of the estimator via flux for *MRAS*. The technique emf for *MRAS* although has if stabilized quickly in a value of speed, distanced of the reference value. Therefore, it is had the techniques through flux and emf without *MRAS*, both not guaranteed the convergence of the speed for the reference value.

The *IM* speed values obtained by experiments and simulations are analyzed to evaluate the estimation algorithm's performance. Speeds come from experimental measurement (tachometer and PSoC algorithm implementation) and model simulation. PSoC implementation main practical issues are described, [2], to establish relationships between proposed indirect measurement system's real time hardware and software.

A. Model Simulations

Implemented algorithm in MATLAB environment. Computational simulation results (voltage, currents and flux in stationary frame) are presented in Figs. 5, 6 and 7. The Fig. 8 illustrates the rotor speed obtained via neural estimator.

B. Experimental Systems

Programmable System on Chip (PSoC) is used in this design for neural estimator implementation, considering PSoC available memory. Simulation and experimental data are obtained by Equacional catalogues and data plate. Besides algorithm development coded in C language dialect of PSoC.

IM main feature is presented in the subsection and in the programmable architecture. **Induction Motor**: 3 phases, voltage: 380 V in Y, frequency: 1700 rpm, hp: 2.25 kW and power factor: 0.82. **Parameters**: L_s=134.5 mH, L_r=76.55 mH, R_r=1.55 Ω and R_s=2.1 Ω.

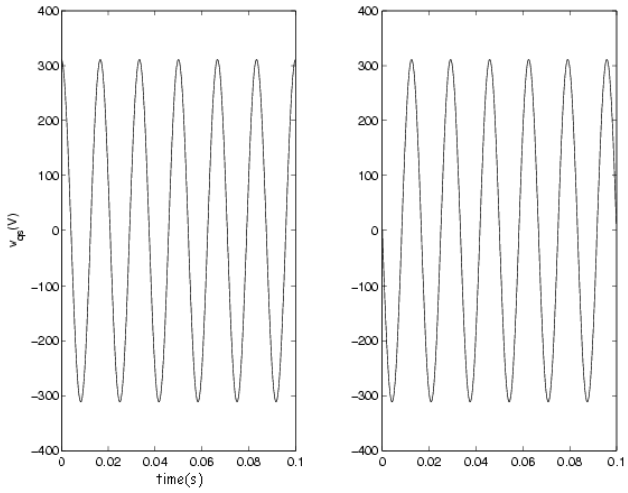

Fig. 5. Voltage in stationary frame obtained from simulation

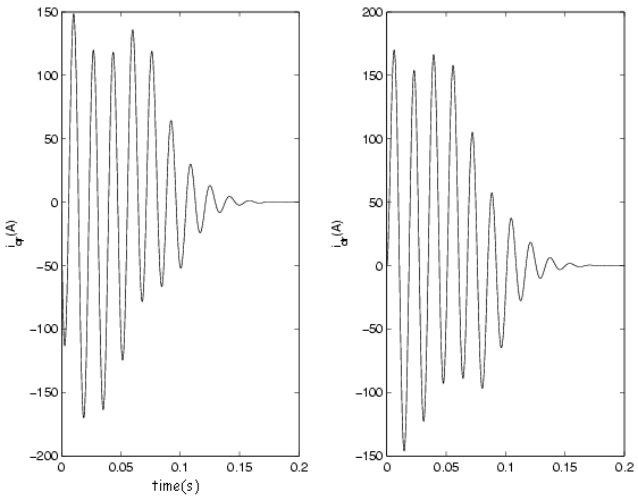

Fig. 6. Currents in stationary frame obtained from simulation

The programmable architecture technology used was developed by PSoC. It's serial specification is CY8C27443. A dedicate software, PSoC Designer version 4.0, is used to perform the block connection in the PSoC. The module used to supply the values of output of the estimated speed is the LCD and the port 0 of the chip is connected to an external display. **Programmable architecture**:M8C Processor Speeds: 24 MHz, Operating voltage: 4.95 V, flash memory used: 8 kB and module: LCD.

C. Analysis

In the regime the estimator neural has good operation, however, it was not possible analyzing the acting in the transitory. The estimation steady state good performance em regime can be checked out from the tachometer measurements. In this case, the tachometer measurements is 1790.00 rpm no load,

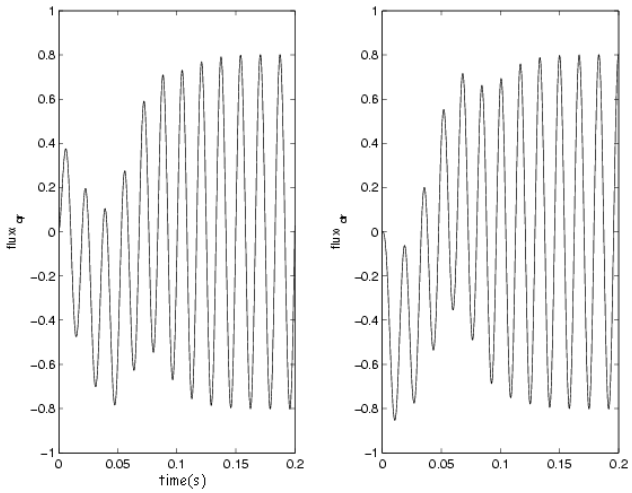

Fig. 7. Flux in stationary frame obtained from simulation

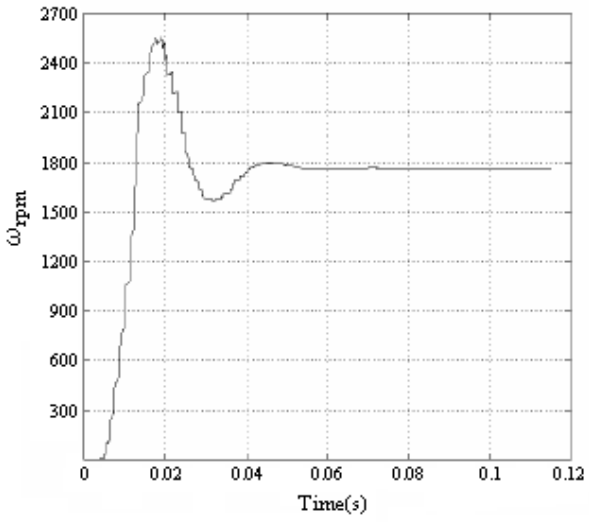

Fig. 8. Rotor speed via neural estimator

model simulation is 1798.80 rpm and display connect the PSoC is 1791.94 rpm.

VI. CONCLUSION

An indirect measurement architecture for *IM* speed estimation has been presented in this paper. This system is based on neural estimator implemented on programmable system on-chip (PSoC), using code C, which executes speed estimator algorithm, considering a 1.0 s time interval for neural estimator.

The results from experiments and simulations indubitably proved the proposed estimator good performance in regime. This efficiency was verified by analysis, considering the following focus: estimator precision when it's compared to conventional speed measurements (tachometer). Speed measurements values from the PSoC and tachometer have shown a smaller than 2% discrepancy.

VII. ACKNOWLEDGEMENTS

The presents investigation was sponsored by PROCAD/CAPES (n^{o}0152/01-3), PRONEX/FAPESQ/CNPq and CEFET-MA.

VIII. REFERENCES

[1] Holtz, Joachim. *Methods for Speed Sensorless Control of AC Drives*. IEEE PCC - Yokohama, pp 415-420, 1993.

[2] Seron, María M., and Braslavsky, Julio H., and Goodwin, Graham C..*Fundamental Limitations in Filtering and Control*. Springer, February, 2004.

[3] Mistry, Sanjay I., and Nair, Satish S.. *Real-time experiments in neural identification and control of disturbance*. Proceedings of the American Control Conference, pp 2307-2311, June 1995.

[4] Irwin, G.W.,Warwick, K., and Hunt, K. J.. *Neural Network Applications In Control*. The Institution of Electrical Engineers, 1 edition, 1995.

[5] Azevedo, F. M., Brasil, L. M., and de Oliveira, R. C. L.. *Redes Neurais com aplicações em Controle e em Sistemas Especialistas*. Visual Books, 1 edition, 2000.

[6] Ben-Brahim, L. *Motor speed identification, via neural network*. IEEE Industry Applications Magazine, 1995.

[7] Ben-Brahim, L. and Tadakuma, S.. *Pratical considerations for sensorless induction motor drive system*. IEEE, 1998.

[8] Elloumi, M., Ben-Brahim, L., and Al-Hamadi, M. A.. *Survey of speed sensorless controls for im drives*. IEEE, 1998.

[9] Ong, Chee-Mun. *Dynamic Simulation of Elestric Machinery*. Prentice Hall

[10] Haykin, S.. *Redes Neurais:princípios e prática*. Ed. Bookman, 2 edition, 2001.

[11] FonsecaNeto, João Viana Da. *Identificação da Velocidade de Motores de Indução Via Rede Neural*,1996.

[12] FonsecaNeto, J. V. da and Mouzinho, L.F.. *Analysis of Speed Estimation Techniques for Asynchronous Machines*. International Association of Science and Technology for Development (Iasted), 2001.

[13] Mouzinho, L.F.. *Redes neurais em estimação de velocidade para máquinas de indução*. Dissertação de Mestrado-UFMA, março, 2003.