# A ROBUST ANN FOR ROTOR SPEED INDIRECT MEASUREMENT OF THE INDUCTION MACHINE

Lucilene F. Mouzinho*, João Viana da Fonseca Neto†, Benedito A. Luciano‡, Raimundo Carlos S. Freire‡

*Departamento de Eletro-Eletrônica
Centro Federal de Educação Tecnológica do Maranhão
São Luís, Maranhão, Brasil

†Departamento de Engenharia Elétrica
Universidade Federal do Maranhão
São Luís, Maranhão, Brasil

‡Departamento de Engenharia Elétrica
Universidade Federal de Campina Grande
Campina Grande, Paraíba, Brasil

Emails: mouzinho@dee.cefet-ma.br, jviana@dee.ufma.br, benedito@dee.ufcg.edu.br, rcsfreire@dee.ufcg.edu.br

**Abstract**— This paper presents a method for the modeling of the rotor speed indirect measurement in an induction machine. It uses an Artificial Neural Network (ANN) algorithm. The algorithm is developed considering the robustness of the induction machine. This method is investigated and realized via environmental *MATLAB* and *C* language. The induction machine model is developed and processed in *MATLAB*, analyzing the behavior of stator voltage and current, rotor flux and rotor speed. These values (stator voltage and current) are storages in a file and, after used with step input signals for the *ANN* algorithm developed in *C* language. The rotor speed is one of the three weights of an *ANN*.

**Keywords**— Modeling of a System, Artificial Neural Networks, Rotor Speed, Induction Machine, Indirect Measurement, Parametric Uncertainty.

## 1 Introduction

State observer models developed to estimate an induction machine speed have as input measured values a stator voltage and current. 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 by electronic devices which perform indirect measurements, contributes a increasing of the induction motor operational robustness, as to the tachometer concern and a reducing of the implementation costs in the control project and the non electric variables supervision.

Due to the temperature, saturation and other nonlinear effects, variations exist in parameters of the induction machine. Several techniques of estimate of parameters in induction machines are used, such as [1], [2], [3]. The variation in the rotor resistance can be of up to 50%, causing a variation in the time constant rotor that influences the flux oriented control [4]. In the robust learning, all the parameters should be learned with the same speed.

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 and specific qualities which justify Artificial Neural Networks, ANN, application in induction machine [5]. Most of real systems present some non linearities and therefore systems linear modeling doesn't represent system 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 systems existence which is linear or whose behavior can be linearized within certain operation area. An *ANN* has a quite promising use in the identification of non linear dynamical systems. *ANN's* can be a proper tool for nonlinear systems modeling due to their learning ability.

## 2 Induction Machine Model

The excitation an *ANN* is realized via stator voltages and currents in a stationary reference frame. The output of an *ANN* is the rotor flux. The induction machine model equations in terms of the vectorial quantities are specified in [6], [7], [8],

$$\vec{v}_s = R_s\vec{i}_s + \dot{\vec{\lambda}}_s, \tag{1}$$

$$0 = R_r\vec{i}_r + \dot{\vec{\lambda}}_r + j\omega_r\vec{\lambda}_r, \tag{2}$$

$$\vec{\lambda}_s = L_s\vec{i}_s + L_m\vec{i}_r, \tag{3}$$

$$\vec{\lambda}_s = L_s\vec{i}_s + \vec{\lambda}_m, \tag{4}$$

$$\vec{\lambda}_r = L_m \vec{i}_s + L_r \vec{i}_r, \tag{5}$$

$$\vec{\lambda}_r = L_r \vec{i}_r + \vec{\lambda}_m, \tag{6}$$

where $\lambda$ is the flux linkage; $L$ the inductance; $v$ the voltage; $R$ the resistance; $i$ the current; $\sigma = 1 - \frac{L_m^2}{L_r L_s}$ the leakage coefficient; $T_r = \frac{L_r}{R_r}$ the rotor time constant; $\omega_r$ the rotor speed and $L_m$ the magnetization inductance. The subscripts $r$ and $s$ represent the rotor and stator reference values, respectively and the subscripts $d$ and $q$ represent $dq$ axis components.

$\vec{v_s} = \left[\ \vec{v}_{ds} \quad \vec{v}_{qs}\ \right]', \quad \vec{i}_s = \left[\ \vec{i}_{ds} \quad \vec{i}_{qs}\ \right]'$

$\dot{\vec{\lambda}}_s = \frac{d\vec{\lambda}_s}{dt}$ induced voltage (time derivative flux).

Considering $\vec{v}_r = \left[\ \vec{v}_{dr} \quad \vec{v}_{qr}\ \right]'$ where 0 (rotor short-circuited) and $\omega = 0$. Similarly as [9], [10], [11] and [12], to verify the robustness of *ANN* the real parameters of the induction machine is considered that presents the following uncertainties (in percentage): $\frac{\Delta R_s}{R_s} = c$, $\frac{\Delta R_r}{R_r} = d$, $\frac{\Delta L_r}{L_r} = e$, $\frac{\Delta L_s}{L_s} = f$, $\frac{\Delta L_m}{L_m} = g$ e $\frac{\Delta\sigma}{\sigma} = h$. The equations (1), (2), (3) and (5) as,

$$\begin{bmatrix} \vec{v}_s \\ 0 \\ \vec{\lambda}_s \\ \vec{\lambda}_r \end{bmatrix} = \begin{bmatrix} (R_s + \Delta R_s) & \frac{d}{dt} & 0 & 0 \\ 0 & 0 & (R_r + \Delta R_r) & \frac{d}{dt} \\ (L_s + \Delta L_s) & 0 & (L_m + \Delta L_m) & 0 \\ (L_m + \Delta L_m) & 0 & (L_r + \Delta L_r) & 0 \end{bmatrix} \begin{bmatrix} \vec{i}_s \\ \vec{\lambda}_s \\ \vec{i}_r \\ \vec{\lambda}_r \end{bmatrix} +$$

$$j\omega_r \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \vec{\lambda}_r. \tag{7}$$

The rotor speed, $\omega_r$, is obtained by rotor flux orientation ($\lambda_r$) [13]. The rotor speed estimation is based on equations (8) and (9). The rotor flux is used for calculation speed [14]. Starting of the Equations (1), (2), (3) and (5), the rotor flux can be written as:

$$\dot{\vec{\lambda}}_d = \frac{(L_r + \Delta L_r)}{(L_m + \Delta L_m)} (\vec{v}_s - (R_s + \Delta R_s)\vec{i}_s - (\sigma + \Delta\sigma)(L_s + \Delta L_s)\dot{\vec{i}}_s) \tag{8}$$

or

$$\dot{\vec{\lambda}}_a = (-\frac{1}{(T_r + \Delta T_r)} + \omega_r j)\vec{\lambda}_r + \frac{(L_m + \Delta L_m)}{(T_r + \Delta T_r)}\vec{i}_s. \tag{9}$$

Developing the equations (8) and (9),

$$\dot{\vec{\lambda}}_d = \frac{L_r}{L_m}(A\vec{v}_s - BR_s\vec{i}_s - C\sigma L_s \dot{\vec{i}}_s), \tag{10}$$

$$\dot{\vec{\lambda}}_a = (-\frac{1}{T_r D} + \omega_r j)\vec{\lambda}_{2r} + \frac{L_m}{T_r} F\vec{i}_s, \tag{11}$$

where $A = \frac{(1+e)}{(1+g)}$, $B = A(1+c)$, $C = A(1+f)(1+h)$, $D = (1 + \frac{e}{d})$ e $F = \frac{(1+g)}{D}$. Where, $\dot{\vec{\lambda}}_d = \frac{d}{dt}\begin{bmatrix} \vec{\lambda}_{dd} \\ \vec{\lambda}_{dq} \end{bmatrix}$ and $\dot{\vec{\lambda}}_a = \frac{d}{dt}\begin{bmatrix} \vec{\lambda}_{ad} \\ \vec{\lambda}_{aq} \end{bmatrix}$, $\dot{\vec{\lambda}}_d$ e $\dot{\vec{\lambda}}_a$ are equivalent expression. The induced voltage, $\dot{\vec{\lambda}}_a$, is a function of $\omega_r$, while $\dot{\vec{\lambda}}_d$ is independent of the $\omega_r$.

The output block that represents the induction machine is given by the desired flux, $\lambda_d$ is obtained by the induced voltage equation (10). In the output model neuronal, the actual flux ($\lambda_a$) is described by equation (11). The flux is independent, expressed by a structure stationary.

## 3 Learning

The learning of the *ANN* is accomplished by back-propagation and supervised algorithm. This type of algorithm uses evens (input, desired output) for, error correction, to adjust the weights of the *ANN*. When the actual flux, $\lambda_a$, approaches the desired flux, $\lambda_d$, the rotor flux of the equation (11) approaches the flux of the equation (10), the *Robust Observer-Neural* (RON) to supply the delayed signals in time for your self input, according to equation (12), [15], [16], [17],

$$y^{maq}(t+1) = f(y^{maq}(t), ..., y^{maq}(t-n+1); u(t), ..., u(t-m+1)). \tag{12}$$

The output system, represented by induction machine, $y^{maq}$, in $t+1$ instant depends of the $n$ passed values of the output system and of $m$ passed values of the $u$ input system, where $(u(t), y^{maq}(t))$ are the evens input/output system values.

The *RON* is used as an estimator to obtain the rotor flux. The error between the reference flux (desired flux), $\lambda_d$, and the estimated flux (actual flux) $\lambda_a$ is used by learning algorithm to adjust weights of the *ANN*. The *ANN* possesses three weights, being two considered as constants.

The weights adjust of the *ANN* is realized via rotor speed that updates $\lambda_a$ until that approaches the values $\lambda_d$. The known the motor parameters, the models *ANN* and induction machine must coincide. However, any difference between the used speed in neural model and speed induction machine can automatically result an error between the outputs of the two estimators. This error between desired flux $\lambda_d$ and actual flux $\lambda_a$ is responsible for the updating the weight in *ANN* model ( rotor speed, $\omega_r$, in equation (9)). This approach is shown in Fig. 1. The back-propagation algorithm is derived the estimator in equation (11) that accompanies the closest possible the estimator in equation (10). To obtain the back-propagation algorithm, the data model patterns of the equation (11) is firstly derived. The rotor flux ($\lambda_r$) instantaneous variation range in $T$ relation to the instant in which $t = Tk$, 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,

$$\dot{\vec{\lambda}}_a(k) = \frac{\vec{\lambda}_a(k) - \vec{\lambda}_a(k-1)}{T}. \tag{13}$$

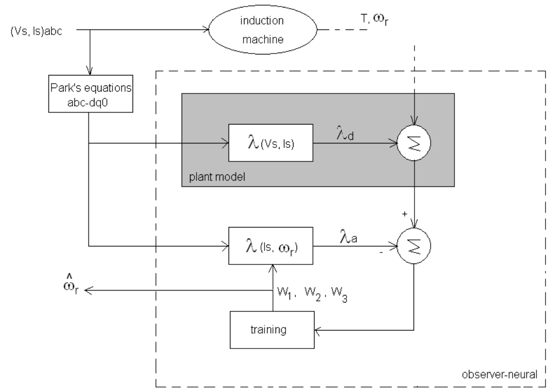


Figure 1: Robust Observer-Neural.

Applying the recursive method in the right of the equation (13), equaling the equation (11), it follows

$$\frac{\vec{\lambda}_a(k)-\vec{\lambda}_a(k-1)}{T} = (\frac{-1}{T_rD}I+\omega_rJ)\vec{\lambda}_a(k-1)+ F\frac{L_m}{T_r}\vec{i}_s(k-1), \quad (14)$$

then

$$\vec{\lambda}_a(k) = I\vec{\lambda}_a(k-1)+(\frac{-1}{T_rD}I+\omega_rJ)T\vec{\lambda}_a(k-1)+ F\frac{L_m}{T_r}T\vec{i}_s(k-1). \quad (15)$$

Organizing terms in relation to matrices $I$, $J$ and variables $\vec{\lambda}_a$ and $\vec{i}_s$,

$$\vec{\lambda}_a(k) = (1-\frac{T}{T_rD})I\vec{\lambda}_a(k-1)+\omega_rTJ\vec{\lambda}_a(k-1)+ F\frac{L_m}{T_r}T\vec{i}_s(k-1). \quad (16)$$

The equation (16) can be written as

$$\vec{\lambda}_a = w_1\vec{x_1}+w_2\vec{x_2}+w_3\vec{x_3} \quad (17)$$

or

$$\vec{\lambda}_a = \sum_{i=1}^{3} w_i\vec{x_i}, \quad (18)$$

with $w_1 = 1-\frac{T}{T_rD}$, $\vec{x_1} = I\vec{\lambda}_a(k-1)$, $w_2 = \omega_rT$, $\vec{x_2} = J\vec{\lambda}_a(k-1)$, $w_3 = F\frac{L_m}{T_rT}$, $\vec{x_3} = I\vec{i}_s(k-1)$, $I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $J = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$ and $T$ is the sampling period.

## 4 Artificial Neural Networks Architecture

The representation of the *ANN* architecture multi-layer perceptron *MLP* in Fig. 2 is composed by two layers; two neurons. In other words, it possesses two outputs that represents the rotor fluxes in the variables $d$ and $q$. The weights of the *ANN* are $w_1$, $w_2$ and $w_3$; $x_1$, $x_2$ and $x_3$ represents the input and the activation function is linear. The

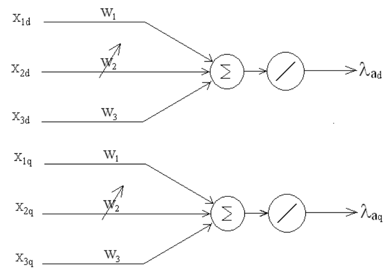


Figure 2: Two neurons with three inputs and two linear outputs.

solution of the *RON* of rotor speed possesses one input layer and one output layer for each one of the two neurons. Then

$$\vec{\lambda}_a = \begin{bmatrix} \lambda_{ad} \\ \lambda_{aq} \end{bmatrix},$$

$$\vec{x} = \begin{bmatrix} x_{1d} & x_{2d} & x_{3d} \\ x_{1q} & x_{2q} & x_{3q} \end{bmatrix}, \quad (19)$$

$$w = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}. \quad (20)$$

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

$$\vec{\varepsilon}(k) = \vec{\lambda}_d(k)-\vec{\lambda}_a(k). \quad (21)$$

The neural network weights $w_1$ and $w_3$ are considered to be constant, therefore only depend on machine parameters. The weight $w_2$ depends on the machine speed and is variable.

The synaptic weights ($w1$, $w2$, $w3$) are adjusted to minimize the energy function [14], [18], [19]. The instantaneous value of the error energy for these neurons is defined as

$$E = \frac{1}{2}\vec{\varepsilon}^{2}(k), \quad (22)$$

or

$$E = \frac{1}{2}\varepsilon^{'}\varepsilon = \frac{1}{2}\|\vec{\lambda}_d(k)-\vec{\lambda}_a(k)\|^2. \quad (23)$$

Amongst neural network weights, only $w_2$ varies. The correction $\Delta w_2(k)$ applied by back-propagation in this weight is:

$$\Delta w_2(k) \propto -\frac{\partial E}{\partial w_2}. \quad (24)$$

By the *Delta Rule*, the gradient $\frac{\partial E}{\partial w_2}$ can be express as [20]:

$$\frac{\partial E}{\partial w_2} = \frac{\partial E}{\partial \varepsilon}\frac{\partial \varepsilon}{\partial x_2}\frac{\partial x_2}{\partial \lambda_a}\frac{\partial \lambda_a}{\partial w_2}, \quad (25)$$

where $\frac{\partial E}{\partial w_2}$ is the sensibility factor that determine the search direction in weights space for synaptic weight $w_2$.

Differentiating of both sides of equation (22) leads to the $\varepsilon$, is

$$\frac{\partial E}{\partial \varepsilon} = \vec{\varepsilon}. \tag{26}$$

Substituting $\vec{\lambda}_a$ a from equation $x_2 = J\vec{\lambda}_a(k-1)$ in equation (21) and differentiating this equation of both sides in relation to $x_2$, we get

$$\frac{\partial \varepsilon}{x_2} = -1. \tag{27}$$

If $x_2 = J\vec{\lambda}_a(k-1)$, then

$$x_2 = f(\vec{\lambda}_a). \tag{28}$$

Differentiating equation (28) in relation $\vec{\lambda}_a$, there is

$$\frac{\partial x_2}{\partial \vec{\lambda}_a} = f'(\vec{\lambda}_a). \tag{29}$$

Differentiating equation (17) in relation $W_2$, leads to

$$\frac{\partial \lambda_a}{\partial w_2} = x_2. \tag{30}$$

Substituting equations (26), (27) and (29) in equation (25),

$$\frac{\partial E}{\partial w_2} = \varepsilon(-1)f'(\vec{\lambda}_a)x_2. \tag{31}$$

Correction $\Delta w_2(k)$ applied to $w_2$, defined as *Delta Rule*, is given by

$$\Delta w_2(k) = -\eta\frac{\partial E}{\partial \vec{\lambda}_a}, \tag{32}$$

where $\eta$ 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 leads to $\varepsilon$.

Substituting equation (31) in (32), is

$$\Delta w_2(k) = -\eta\varepsilon(-1)f'(\vec{\lambda}_a)x_2, \tag{33}$$

or

$$\Delta w_2(k) = -\eta\delta(k)x_2, \tag{34}$$

where $\delta(k)$ local gradient, as

$$\delta(k) = -\frac{\partial E}{\partial \vec{\lambda}_a}. \tag{35}$$

Differentiating equation (23) in relation to $\vec{\lambda}_a$ gives us

$$\frac{\partial E}{\partial \vec{\lambda}_a} = \frac{1}{2}\frac{\partial[(\vec{\lambda}_d(k) - \vec{\lambda}_a(k))'(\vec{\lambda}_d(k) - \vec{\lambda}_a(k))]}{\partial \vec{\lambda}_a(k)}, \tag{36}$$

$$\frac{\partial E}{\partial \vec{\lambda}_a} = \frac{1}{2}\frac{\partial[\vec{\lambda}_d(k)'\vec{\lambda}_d(k) - \vec{\lambda}_d(k)'\vec{\lambda}_a(k) - \vec{\lambda}_a(k)'\vec{\lambda}_d(k) + \vec{\lambda}_a(k)'\vec{\lambda}_a(k)]}{\partial \vec{\lambda}_a(k)} \tag{37}$$

as $\vec{\lambda}_d(k)'\vec{\lambda}_a(k) = \vec{\lambda}_a(k)'\vec{\lambda}_d(k)$ then,

$$\frac{\partial E}{\partial \vec{\lambda}_a} = \frac{1}{2}\frac{\partial[(\vec{\lambda}_d(k)^2)' - (2\vec{\lambda}_d(k))'\vec{\lambda}_a(k) + (\vec{\lambda}_a(k)^2)']}{\partial \vec{\lambda}_a(k)} \tag{38}$$

deriving,

$$\frac{\partial E}{\partial \vec{\lambda}_a} = \frac{1}{2}(-2\vec{\lambda}_d(k)' + 2\vec{\lambda}_a(k)'), \tag{39}$$

$$\frac{\partial E}{\partial \vec{\lambda}_a} = (-\vec{\lambda}_d(k)' + \vec{\lambda}_a(k)'), \tag{40}$$

substitute the equation (40) in (35),

$$\delta(k) = (\vec{\lambda}_a - \vec{\lambda}_d)'. \tag{41}$$

Substitute the equation (35) and $x_2 = J\vec{\lambda}_a(k-1)$ in equation (34),

$$\Delta w_2 = -\eta(\vec{\lambda}_a(k) - \vec{\lambda}_d(k))'J\vec{\lambda}_a(k-1). \tag{42}$$

Starting of the *Delta Rule* the new weight is,

$$w_2(k) = w_2(k-1) + \eta\Delta w_2(k), \tag{43}$$

where $\eta$ is learning coefficient and $k$ is incremented of the 1 for each sweeping through input-output set.

The *Delta Rule* in equation (34) is modified to increase the learning range without oscillations including itself a moment term, as equation (44).

$$\Delta w_2(k) = -\eta\delta(k)x_2 + \alpha\Delta w_2(k-1). \tag{44}$$

The $\alpha$ coefficient the moment constant, determines previous weight modification effects in the actual weight. However, it is better to use equation (44) instead of equation (43).

Knowing $w_2 = \omega_r T$, then $\omega_r = \frac{w_2}{T}$, varying $\omega_r$,

$$\Delta\omega_r = \frac{\Delta w_2}{T}. \tag{45}$$

And, finally the substitution of the equation (44) in equation (45), delivers the rotor speed estimate which is given by,

$$\Delta\omega_r = -\frac{1}{T}\eta\delta(k)X_2 + \frac{1}{T}\alpha\Delta w_2(k-1), \tag{46}$$

or

$$\hat{\omega}_r(k) - \hat{\omega}_r(k-1) = -\frac{1}{T}\eta\delta(k)x_2 + \frac{1}{T}\alpha\Delta w_2(k-1). \tag{47}$$

Then, the estimated speed is [22]

$$\hat{\omega}_r(k) = \hat{\omega}_r(k-1) - \frac{1}{T}\eta\delta(k)x_2 + \frac{1}{T}\alpha\Delta w_2(k-1). \tag{48}$$

Based on Liou, [21], and combining the equations (17) and (46), we get

$$(\Delta w_k, y_a(k)) = F(w_k, x_k, y_d(k)), \tag{49}$$

where $F$ utilizes the weights $w_k$, the input $x_k$ and the desired value $y_d$ as input; $\Delta w_k$ and $y_a(k)$ as output. In Fig. 3 the flowchart of the back-propagation to train the $ANN$ is shown. In this algorithm, the output of the $ANN$ doesn't supply the value of the quantity of interest. The desired quantity is obtained by the variation of the weight. The rotor speed is extracted of the weight $w_2$, as it is verified in equation (48) and shown in Fig. 1. The frequency of adjustments of the weight $w_2$ for the backpropagation algorithm is accomplished on-line.

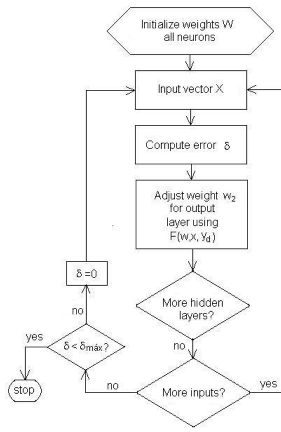


Figure 3: Back-propagation training flowchart

We implemented the algorithms in a $MATLAB$ environment. The behavior of the stator voltage/currents and rotor flux are shown in Figs. 4, 5 and 6. In [22] is used the same development of the models of a induction machine and obtains the rotor speed via $ANN$ without robustness as show in Fig. 7.

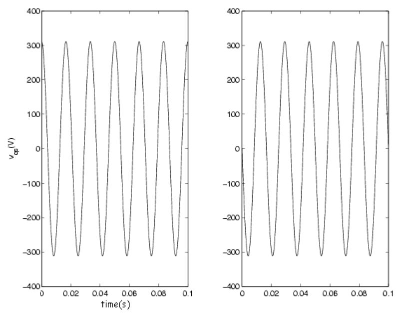


Figure 4: Stator voltage in a stationary frame

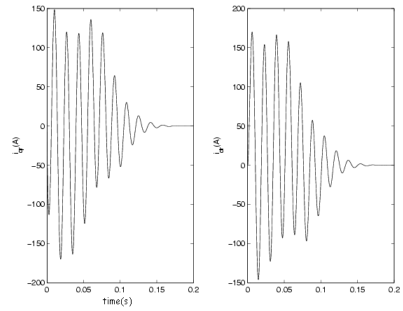


Figure 5: Stator current in a stationary frame

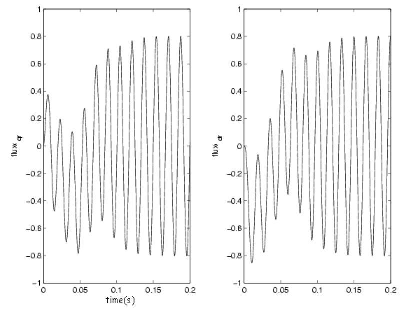


Figure 6: Rotor flux in a stationary frame

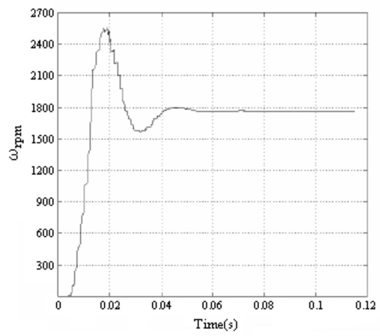


Figure 7: Rotor speed via $ANN$

## 5 Computational Results

Computational data are obtained by Equacional catalogues and data plaque of the induction machine. The algorithm was $ANN$ also developed coded in *Builder C* language version 4.5. The main feature of the induction machine are presented: **Induction Motor**: 3 phases, voltage: 380 $V$ in $Y$, speed: 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 $\Omega$ and $R_s$=2.1 $\Omega$.

In this case, the size of the variables influences the processing time of the algorithm in $C$ language, due to the use of matrices. Some strategies were used to accelerate the method so that the execution of the algorithm was favorable, allowing that matrices to pretend to have a smaller number of points.

The neuronal algorithm core was developed. A set of the stator voltages and currents samples in reference $dq0$ it was stored in a table and considered initially constant.

The output signal of the $ANN$ is through a linear activation function. For practical purposes, the $ANN$ is considered non linear due the operational limits of the induction machine. If the $ANN$ is working on a linear operational range the neural observer is considered linear.

The essential difference between this work and the published works in [18], [19] and [22] was the rotor speed indirect measurement via $ANN$, included the parametric uncertainties in the induction machine model. In the works mentioned previously was not considered of the robustness of the induction machine.

An analysis of performance of the use of $ANN$ without robustness was accomplished [22]. The good performance of the estimator in state space was verified by comparison with the tachometer values in permanent regime. In this case, the tachometer delivered a value of 1790.00 $rpm$, considering the unloaded motor. The speed obtained for simulation without considering the robustness was 1798.80 $rpm$.

## 6 Conclusion and Remarks

A proposed model for determination of the rotor speed in induction machine has been presented in this paper. This model is based on a neural observer. The neural algorithm was developed in $MATLAB$ and $C$ language.

The calculated results proved the proposed estimator a good performance. The efficiency of model was verified by an analysis without robustness, considering the following focus: estimator precision when it's compared to conventional speed measurements (tachometer).

This method proposed for indirect measurement of speed in machine via $ANN$ is enough promising and therefore implementations of these algorithms in systems embedded tend to substitute in large scale the electromechanical devices for direct measurement of quantity of the induction machine, such as: angular position, speed and torque.

## 7 Acknowledgements

The present investigation was sponsored by PROCAD/CAPES (n$^\sharp$0152/01-3), FAPEMA and CNPq.

## References

[1] Jennifer Stephan, Marc Bodson and John Chiasson. Real-time estimation of the parameters and fluxes of induction motors. *IEEE Transactions on Industry Applications*, vol. 3, pages 746-759, May/June 1994.

[2] D. S. Oh, K. Y. Cho and M. J. Youn. New rotor parameter estimation for a lux and speed control of induction machine considering saturation effects. *Industrial Electronics Control and Instrumentation-IECON*, vol.1, pages 561-566, 1991.

[3] Shen-Shu Xiong and Zhao-Ying Zhou. Dynamic parameter estimation of velocity sensors using an indirect measurement approach. *IEEE Instrumentation and Measurement*, pages 794-797, May 2001.

[4] Prashant Mehrotra, John E. Quaicoe, and R. Venkatesan. Development of an artificial neural network based induction motor speed estimator. *IEEE Transaction on Industrial Electronics*, pages 682-689, 1996.

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

[6] Lazhar Ben-Brahim and Susumu Tadakuma, Pratical considerations for sensorless induction motor drive system. *IEEE*, 1998.

[7] Paul C. Krause, *Analysis of Electric Machinery* (New York, 10 edition, 1986).

[8] Chee-Mun Ong. *Dynamic Simulation of Electric Machinery* ( Prentice Hall PTR, 10 edition, 1998).

[9] Pompeo Marino, Michele Milano and Francesco Vasca. Robust neural network observer for induction motor control. *IEEE Transaction on Industrial Electronics*, pages 699-705, 1997.

[10] Pompeo Marino, Michele Milano and Francesco Vasca. Linear quadratic state feedback and robust neural estimator for field oriented controlled induction motors. *IEEE Transaction on Industrial Electronics*, pages 150-161, February 1999.

[11] D. Fodor, G. Griva and F. Profumo. Compensation of parameters variations in induction motor drives using a neural network. *Power Electronics Specialists Conference PESC '95 Record*. 26th Annual IEEE, pages 1307-1311, June 1995.

[12] D.-W. Gu, P.Hr. Petkov and M.M Konstantinov. Robust Control Design with MATLAB. Springer-Verlag, 2005.

[13] Ion Boldea and S.A. Nasar. *Electric Drives.* CRC press, 1998.

[14] Mohsen Elloumi, Lazhar Ben-Brahim and Mohamed A Al-Hamadi. Survey of speed sensorless controls for in drives. *Industrial Electronics Society. IECON '98. Proceedings of the 24th Annual Conference of the IEEE* , Aachen-Germany, pages 1018-1023, vol. 2, 1998.

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

[16] James Ting-Ho Lo. Robust adaptive identification of dynamic systems by neural networks. *IEEE Transaction on Industrial Electronics*, pages 1121-1126, 1997.

[17] M.T. Wishart and R.G. Harley. Identification and control of induction machines using artificial neural networks. *IEEE Transactions on Industry Applications*, pages 612-619, May-June 95.

[18] J.V. Fonseca Neto, L.F. Mouzinho, B.A. Luciano and R.C.S. Freire. Aplicações de redes neuronais artificiais e medição indireta de velocidade via observador-neuronal em máquinas de indução. *VII SBAI - VII Brazilian Symposium on Intelligent Automation*, São Luís-MA, Brazil, pages 1-8, September 2005.

[19] L.F. Mouzinho, J.V. da Fonseca Neto, B.A. Luciano and R.C.S. Freire. Induction Machine Speed Neural Estimator Implemented in Programmable Architecture. *Congresso Brasileiro de Redes Neurais*, Natal, Rio Grande do Norte, Brasil, 2005.

[20] Sun Yuang Kung. *Digital Neural Networks.* PTR Prentice Hall, New Jersey, 10 edition, 1993.

[21] Cheng-Yuan Liou and Yen-Ting Kuo. Data Flow Design for the Backpropagation Algorithm. *National Science Council*, pages 1-6, 2002.

[22] L.F. Mouzinho, J.V. da Fonseca Neto, B.A. Luciano and R.C.S. Freire. Induction Machine Neural Estimator using Embedded System. *IEEE (Institute of Electrical and Electronics Engineers), IMTC (Instrumentation and Measurement Technology Conference)*, Ottawa, Canada, May, pages 1291-1296, 2005.