COMPARATIVE STUDY ON FLUX OBSERVERS FOR INDUCTION MOTOR DRIVES

Ademir Nied
Seleme I. S. Junior
Benjamim R. de Menezes
Gustavo G. Parma
Júlio C. G. Justino
Universidade Federal de Minas Gerais
Av. Antônio Carlos 6627
Belo Horizonte – MG – CEP 31270-010
{nied,seleme,brm,parma,julio}@cpdee.ufmg.br

***Abstract* – This paper presents a comparative study between two traditional flux observers and two neural flux observers. The neural network topology is a standard multilayer perceptron network, and the two on-line training algorithms are based on Sliding Mode Control (SMC) theory. The stator flux neural observers present a better performance with respect to rotor resistance uncertainties whereas the traditional ones have a better response when the uncertainties are with the stator resistance.**

KEYWORDS

Flux Observers, Sliding Mode Control, Artificial Neural Networks, Flux Neural Observer.

I. INTRODUCTION

It is quite known that the precise magnetic flux estimation is crucial for the implementation of the various approaches of direct field oriented control for induction motors (IM). Several methods for the flux acquisition have been proposed. It is usual to classify them as either Flux Estimators or Flux Observers [1], [2].

Consider a linear multivariable system modeled in the State Space as:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) + \boldsymbol{D}\boldsymbol{\omega}(t) \quad (1)$$

$$\boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{H}\boldsymbol{\omega}(t) \quad (2)$$

where, $\boldsymbol{x}(t)$ e $\dot{\boldsymbol{x}}(t)$ are, respectively, the n-dimensional state vector and its derivative, $\boldsymbol{u}(t)$ is the m-dimensional vector of the known inputs and $\boldsymbol{\omega}(t)$ is the k-dimensional vector of the unknown or unavailable inputs, representing the external disturbances and the parametric uncertainties. Eq. (2) is the measured outputs equation of the system, where $\boldsymbol{y}(t)$ is the p-dimensional vector of these outputs. $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, $\boldsymbol{D}$ and $\boldsymbol{H}$ are matrices of appropriate dimensions considered as known.

Thus, the estimation process consists in generating state $\boldsymbol{x}(t)$ from the known or available input $\boldsymbol{u}(t)$, and from the output, $\boldsymbol{y}(t)$, given the matrices $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, $\boldsymbol{D}$ and $\boldsymbol{H}$. In simple words, the estimation is a sort of real time simulation of the equations which govern the system dynamics, in order to obtain the state variables which, for some reason, are not available to measure. It is also true that this simulation is in open loop.

The estimator performance can be ameliorated by using both the input, $\boldsymbol{u}(t)$, and the output, $\boldsymbol{y}(t)$ in (1) and (2). Therefore, an observer can be obtained by using a predicted error correction term in this *real time simulation* (estimation), which is a function of the difference of the predicted output, $\hat{\boldsymbol{y}}(t) = \boldsymbol{C}\hat{\boldsymbol{x}}(t)$, and the real (measured) one, $\boldsymbol{y}(t)$. The observer resulting from this procedure is described as:

$$\dot{\hat{\boldsymbol{x}}}(t) = \boldsymbol{A}\hat{\boldsymbol{x}}(t) + \boldsymbol{B}\boldsymbol{u}(t) + \boldsymbol{D}\boldsymbol{\omega}(t) + \boldsymbol{L}[\boldsymbol{C}\hat{\boldsymbol{x}}(t) - \boldsymbol{y}(t)] \quad (3)$$

$$\boldsymbol{y}(t) = \boldsymbol{C}\hat{\boldsymbol{x}}(t) + \boldsymbol{H}\boldsymbol{\omega}(t) \quad (4)$$

where $\boldsymbol{L}$ represents the gain matrix of the observer, whereas, the term within the brackets is the prediction error correction term. The prediction error dynamics is obtained by putting (3) into (1), as:

$$\dot{\boldsymbol{e}}(t) = (\boldsymbol{A} + \boldsymbol{L}\boldsymbol{C})\boldsymbol{e}(t) - (\boldsymbol{D} + \boldsymbol{L}\boldsymbol{H})\boldsymbol{\omega}(t) \quad (5)$$

Notice that when $\boldsymbol{L}=\boldsymbol{0}$, the observer becomes an estimator. The error dynamics is given by the eigenvalues of the matrix $(\boldsymbol{A}+\boldsymbol{L}\boldsymbol{C})$. If this system is observable, by choosing appropriately the values of $\boldsymbol{L}$, the eigenvalues of $(\boldsymbol{A}+\boldsymbol{L}\boldsymbol{C})$ can be arbitrarily chosen, thus imposing arbitrarily the error dynamics. In the presence of uncertainties ($\boldsymbol{w}(t)\neq\boldsymbol{0}$), high values of the gains in $\boldsymbol{L}$, chosen in order to assure a fast convergence of the observed error, have also an impact on the effect of the disturbances, given that the eigenvalues of the matrix $(\boldsymbol{D}+\boldsymbol{L}\boldsymbol{H})$ are also changed with $\boldsymbol{L}$. Therefore, there is a tradeoff between a fast convergence and a low sensibility parameter uncertainties and external disturbances.

The observer is an estimator in closed loop, using the input signals and a feedback signal obtained from the system output and the process dynamic model. The prediction error correction term allows a faster tracking capability then that of the corresponding estimator whose dynamics is dictated by the natural dynamics of the system. There are several schemes of observers proposed in the literature. In [2], ten different topologies of observers are presented, using different structures for the correction term.

The issue of flux observers, robust with respect to parameter variation, capable of a fast convergence is still

lacking some effort of research. In this sense, Artificial Neural Networks (ANN) is a valid alternative. By training the ANN adequately, using the right inputs, it is possible to predict the unavailable variable, the stator flux in our case, in a robust way.

The paper is organized as follows: Section II describes the stator flux neural observer whereas Section III presents the on-line training algorithms proposed by [3] and the adapted algorithm with the simplifications based on [4]. Section IV shows a comparison of simulation results between conventional observers (Gopinath and Luenberger) and neural observers. Finally, Section V presents the conclusions.

II. STATOR FLUX NEURAL OBSERVER

The most popular algorithm for training multi-layer ANN based on MCP node [5] is back-propagation [6]. The training made by adjusting the weights of the ANN through the gradient method, aiming at the minimization of the cost function (error) of the system. On-line training algorithms have to be able to adapt the ANN parameters as a function of parameter variations occurred in the plant, thus allowing a better modeling of the system. According to [7], on-line training algorithms, based on the Sliding Mode Control theory [8], present high speed of convergence and robustness.

The induction motor can be modeled in an orthogonal reference (α,β-frame), disregarding the homopolar component, by the following equation set:

$$v_{s\alpha} = R_s i_{s\alpha} + \frac{d\varphi_{s\alpha}}{dt} \quad (6)$$

$$v_{s\beta} = R_s i_{s\beta} + \frac{d\varphi_{s\beta}}{dt} \quad (7)$$

where:

R_s: stator resistance;

$v_{s\alpha}$, $v_{s\beta}$: stator voltages referred to the (α,β-frame);

$i_{s\alpha}$, $i_{s\beta}$: stator current (α,β-frame);

$\varphi_{s\alpha}$, $\varphi_{s\beta}$: stator flux (α,β-frame).

Consider Eqs. (6) and (7) above. The stator flux components can be derived using the stator current and voltage. The α and β stator current components are used as the input to the ANN, having the α and β components of the stator flux as the output. The α and β components of the voltage are used for the ANN on-line training, as depicted in Fig. 1.

The ANN used has two layers, with 2 inputs, 5 nodes of the hidden layer and 2 outputs. The number of nodes of the hidden layer was determined by the analysis of the simulation results which minimizes the computational burden as the number of nodes varies, provided that the flux estimation is not compromised.

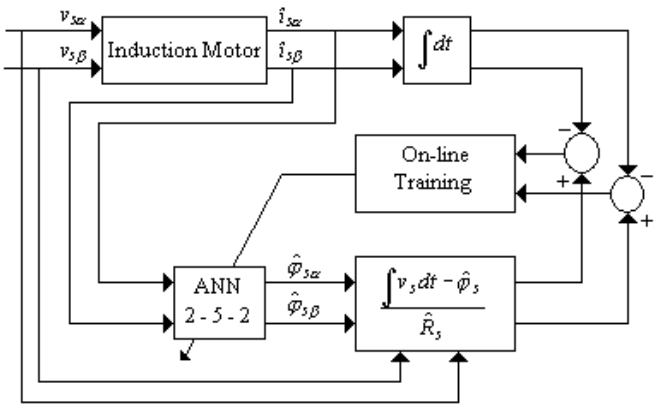

Fig. 1: Block diagram of the neural stator flux observer

Notice that the observer is based one parameter only, the stator resistance, which doesn't vary too much and can be easily obtained by simple laboratory tests.

III. TRAINING ALGORITHMS

The development of the algorithms in this section is based on the ANN structure shown in Fig. 2, in which:

n: inputs;

m: nodes of the hidden layer;

p: outputs;

$\boldsymbol{T}$: input vector with bias;

$\boldsymbol{Y_H}$: output vector of the hidden layer, with bias;

$\boldsymbol{Y}$: output vector of the ANN;

$\boldsymbol{Z}$: weight matrix which connects the input to the hidden layer nodes, with dimension m × (n+1), being Z_{ih} the weight which connects the input h (n≥h) to the input of the node i (m≥i) of the hidden layer;

$\boldsymbol{W}$: weight matrix which connects the output to the hidden layer nodes, with dimension p × (m+1), being W_{ih} the weight which connects the output i (n≥h) to the output of the node j of the hidden layer;

$f_H(.)$: activation function of the hidden layer nodes, using the tangh function;

$f(.)$:activation function of the output layer nodes, using the tangh function;

$f'_H(.)$: derivative of the activation function of hidden layer nodes related to the weights;

$f'(.)$:derivative of the activation function of output layer nodes related to the weights;

It can be seen from Figure 2 that:

$$\boldsymbol{Y_H} = f_H(\boldsymbol{R}) \quad (8)$$

$$\boldsymbol{Y} = f(\boldsymbol{V}) \quad (9)$$

$$\boldsymbol{R} = \boldsymbol{Z.T} \quad (10)$$

$$\boldsymbol{V} = \boldsymbol{W.T} \quad (11)$$

where $\boldsymbol{R}$ is the linear output of the hidden layer and $\boldsymbol{V}$ is the linear output of the output layer, *i.e.*,

$$R_i = \sum_{k=1}^{n+1} Z_{ik} T_{ik} \quad (12)$$

$$V_j = \sum_{k=1}^{m-1} W_{jk} Y_{Hk} \quad (13)$$

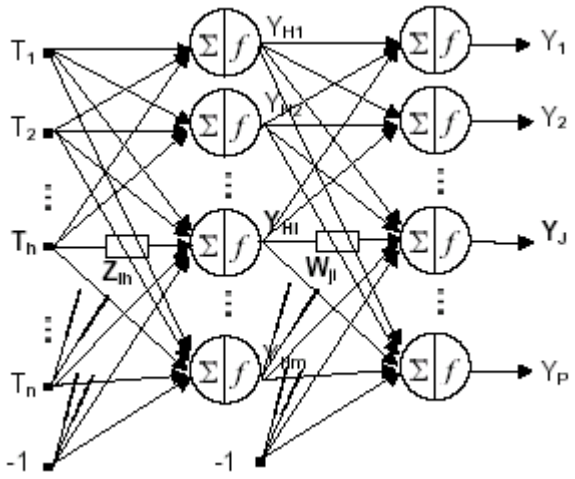

Fig. 2: ANN used as a stator flux observer.

The on-line training algorithms for the ANNs based in SMC are as follows:

A. Training algorithm according to Parma [3]

For each new node of the output layer there is a sliding surface, defined as:

$$S_j = X_{2j} + CX_{1j} \quad (14)$$

$$X_{1j} = (Y_{dj} - Y_j) f'(V_j) \quad (15)$$

$$X_{2j} = \frac{\partial X_{1j}}{\partial t} \quad (16)$$

where Y_{dj} is the j-th desired output for the output layer for j = 1,2,...,p.

For each node of the hidden layer, the following sliding surface is defined:

$$S_{Hi} = X_{2HI} + C_H X_{1Hi} \quad (17)$$

$$X_{1Hi} = f'_H(R_i) \sum_{j=1}^{p} [X_{1j} W_{ji}] \quad (18)$$

$$X_{2Hi} = \frac{\partial X_{1Hi}}{\partial t} \quad (19)$$

Consider Eqs. (15) and (18). The following rules for the updates of the weights are defined:

$$\dot{W}_{ji} = \alpha . sign(S_j) |X_{1j}| Y_{Hi} \quad (20)$$

$$\dot{Z}_{ih} = \beta . sign(S_{Hi}) |X_{1Hi}| T_h \quad (21)$$

Once the sliding surface is defined, it is necessary to determine the limits for gains α and β such that the existence of sliding modes is guarantied. Therefore, the following conditions have to be assured:

- T_h (h=1,...,n) and Y_{dj} (j=1,...,p) are bounded with bounded derivatives;
- $f_H(.)$ and $f(.)$ are bounded with bounded time derivatives.

According to [8], the existence of sliding modes is assured when $S.\dot{S} \le 0$, being $\dot{S}$ the time derivative of S. The deduction of the bounds on α and β are shown in [3]. In order to make the sliding on the surface smoother, the following gains were used: $\alpha=\beta=1e^5$. With such values, chattering problems are avoided.

B. Adapted algorithm

Martens and Weymaere present in [4] an equalized error backpropagation algorithm for the on-line training of multilayer perceprons. The training is done by adjusting the output of the ANN by the gradient method, aiming at minimizing the cost function (of error) of the system output. This proposition is summarized by the following equation:

$$\Delta y_i = -\eta \frac{\partial \varepsilon}{\partial y_i}, \quad i = 1...p \quad (22)$$

Considering Eq. (22), Eqs. (15) and (18) can be written as:

$$X_{1j} = (Y_{dj} - Y_j) \quad (23)$$

$$X_{1Hi} = \sum_{j=1}^{p} [X_{1j} W_{ji}] \quad (24)$$

The rest of the equations derived in [3] remain the same in this approach. As for the bounds for α and β, the same values were adopted, *i.e*, $\alpha=\beta=1.e^5$.

IV. SIMULATION RESULTS

A program written in C was developed in order to compare the stator/rotor flux average percentage error of the two algorithms presented in the previous section with the two conventional Gopinath and Luenberger observers.

The conventional observers use the DUFOR (*Direct Universal Field Oriented Rotor*) as a controller for the simulation and implementation whereas the neural observers use the UFOVS (*Universal Field Oriented Stator*). It is noted that no special care was taken when selecting the control gains. The main concern being the observers performance. The numeric integration of the model differential equations is performed using a forth-order Runge-Kutta method.

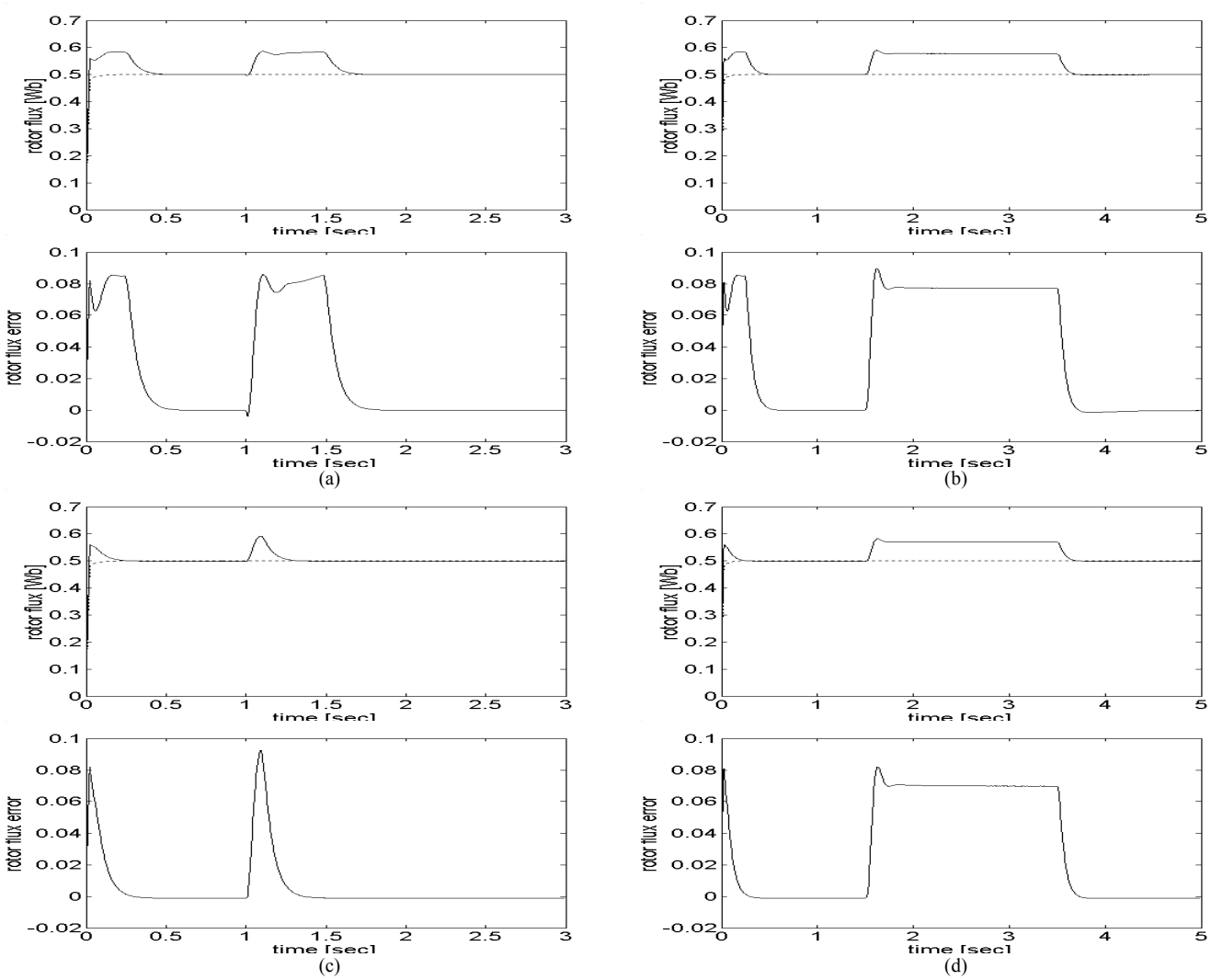

Fig. 3: Gopinath's simulation results with 20% increase of the rotor resistance: (a) Rotor flux and rotor flux error during motor start/speed reversion at 150 elec.rad/sec. (b) Rotor flux and rotor flux error during application/rejection of load at 150 elec.rad/sec. (c) Rotor flux and rotor flux error during start/speed reversion at 30 elec.rad/sec. (d) Rotor flux and rotor flux error during application/rejection of load at 30 elec.rad/sec.

The induction motor model parameters are presented in Table I, whereas the simulation parameters are shown in Table II.

In these simulations, the induction motor was submitted to the following transients:

- motor start and speed reversion (with no load);
- application and rejection of load.

The above transients are done under the following conditions of parametric uncertainty and speed:

- 20% increase of the stator resistance with motor speed of 150 elec.rad/sec;
- 20% increase of the rotor resistance with motor speed of 30 elec.rad/sec.

The weights of the ANN are initialized when starting the motor through a sampling of normal distribution with zero mean. After starting, the weights are updated with the output values obtained at the last simulation step.

Some simulation results are shown in Figs. 3 and 4. Figure 3 shows the transients of the Gopinath's algorithm with 20% increase of the rotor resistance. Figure 4 shows the transients of the Adapted algorithm with 20% increase of de stator resistance.

Those figures show the worst case for both observers: Gopinath and Adapted. These worst cases correspondent to: rotor resistance uncertainty for Gopinath, and stator resistance uncertainty for Adapted. It can be seen that the Adapted neural observer presents smaller flux error then that obtained by Gopinath observer for rotor resistance variation. The opposite happens when the stator resistance varies. Nevertheless, the Adapted neural observer is less sensitive for variations in stator resistance variation then Gopinath is sensitive to rotor resistance drift. This can be verified observing the results of Tables III and IV which synthesize the results obtained by simulation.

The dynamic behavior of both torque and speed follow those of the flux, given that the motor currents are measured. Notice also that both neural observers present similar results. Nevertheless, the adapted algorithm approach has the advantage of being simpler and less time consuming.

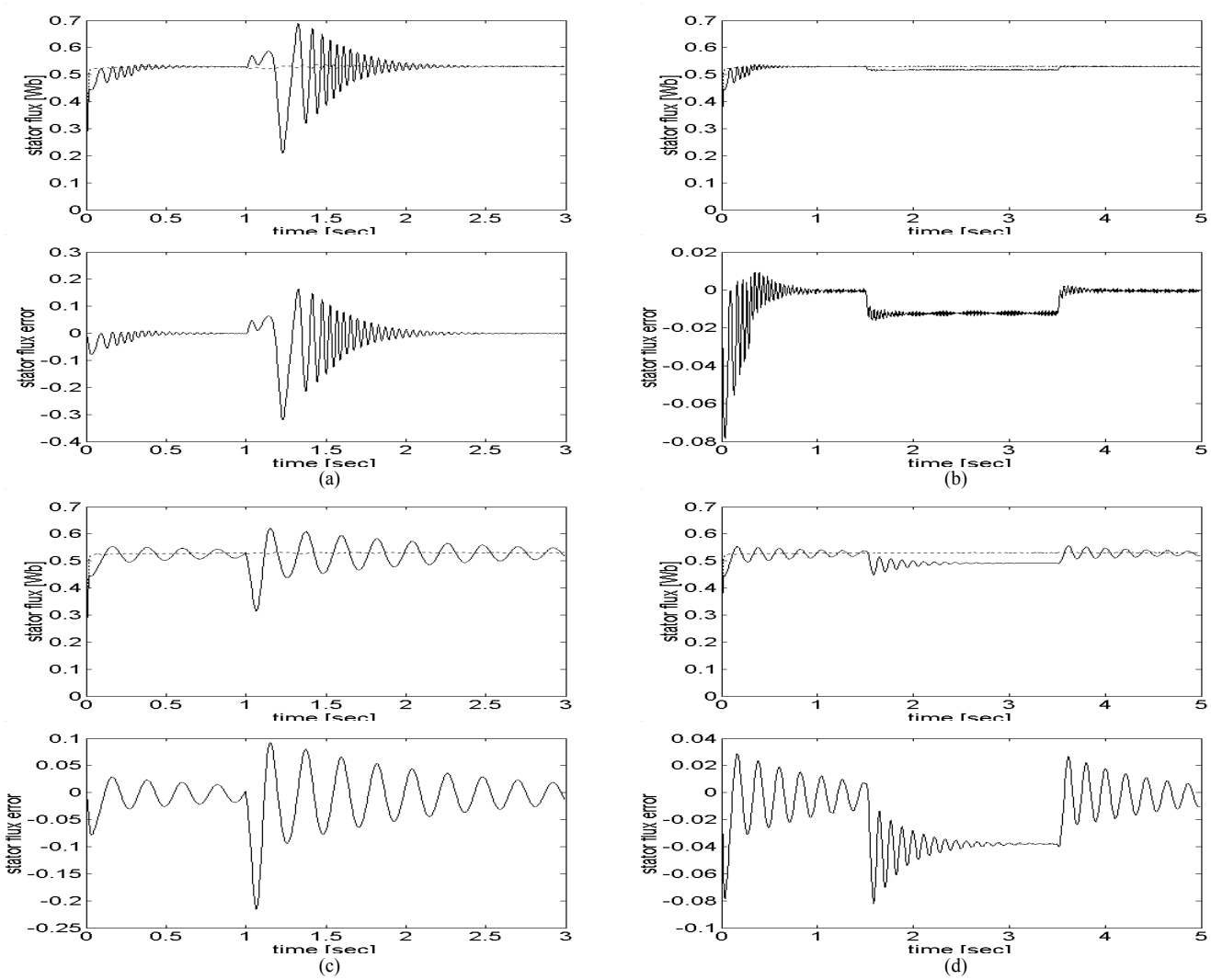

Fig. 4: Adapted algorithm's simulation results with 20% increase of the stator resistance: (a) Stator flux and stator flux error during motor start/speed reversion at 150 elec.rad/sec. (b) Stator flux and stator flux error during application/rejection of load at 150 elec.rad/sec. (c) Stator flux and stator flux error during start/speed reversion at 30 elec.rad/sec. (d) Stator flux and stator flux error during application/rejection of load at 30 elec.rad/sec.

TABLE I
Induction Motor parameters

Power (HP)	2
Rated phase voltage (V)	220
Speed (rpm)	1720
Stator resistance (Ω)	4.08
Rotor resistance (Ω)	4.87
Stator leakage Inductance (H)	0.3154
Rotor leakage Inductance (H)	0.3235
Magnetizing inductance (H)	0.305
Rotational loss coefficient ($W.s^2/rad^2$)	0.018

TABLE II
Parameters used for the simulation

Integration step (μs)	1
Simulation time (s)	3[1] e 5[2]
Sampling frequency (kHz)	4
Voltage at the DC link (V)	300
Load used for the transients (Nm)	4
Reference speed of the motor (elec.rad/s)	150
Reference flux (Wb) of neural observers	0.53
Reference flux (Wb) of conventional observers	0.50

[1] *Simulation time for starting and speed reversion.*
[2] *Simulation time for loading/unloading the motor.*

TABLE III
Flux average percentage error with 20% increase of the stator resistance

	Start/speed reversion		Application/rejection of load	
	w_r = 150 elec.rad/s	w_r = 30 elec.rad/s	w_r = 150 elec.rad/s	w_r = 30 elec.rad/s
Gopinath	0.02	0.09	0.36	0.07
Luenberger with vs[3]	0.02	0.09	0.37	0.07
Parma	0.9	0.91	0.67	1.79
Adapted	0.9	0.91	0.69	1.79

[3] *In this observer the estimator is based on the current model whereas the correction term is derived from the voltage model.*

TABLE IV
Flux average percentage error with 20% increase of the rotor resistance

	Start/speed reversion		Application/rejection of load	
	w_r = 150 elec.rad/s	w_r = 30 elec.rad/s	w_r = 150 elec.rad/s	w_r = 30 elec.rad/s
Gopinath	2.18	0.51	3.6	2.92
Luenberger with vs	2.18	0.5	3.6	2.92
Parma	0.03	0.0	0.03	0.0
Adapted	0.01	0.0	0.01	0.0

V. CONCLUSIONS

A new neural network based observer for the stator flux of induction motor has been presented. Simulation results of this adapted algorithm and of the former one proposed by Parma et al. [3] have been shown.

It can be seen that the much simpler adapted algorithm proposed has the same good performance as the original one. Nevertheless, the adapted algorithm approach has the advantage of being simpler and less time consuming.

The neural network based observers are compared with traditional Gopinath and Luenberger observers. It can be verified from simulation results that the neural observers of the stator flux present a flux average percentage error (considering both case – 20% increase of rotor and stator resistance) which is smaller than those for the traditional observers.

Finally, one notes that:

1) The neural observers are depended on one electrical parameter only (these observers are insensitive to rotor resistance variations), the stator resistance, which doesn't vary too much and can be easily obtained by simple laboratory tests. The traditional observers are depended on two parameters: rotor and stator resistance;

2) The neural observers' algorithms do not depend on the motor speed measurement, which is a great advantage nowadays, considering the industrial induction motor drives trend.

ACKNOWLEDGEMENT

The autors are grateful to CAPES for the financial support for this investigation.

REFERENCES

[1] G. C. Verghese and S. R. Sanders, "Observers for flux estimation in induction machines," *IEEE Trans. on Industrial Electronics.*, vol. 35, no. 1, February 1988.

[2] Y. Hori, V. Cotter and Y. Kaya, "A novel induction machine flux observer and its application to a high performance AC drive system," *Proceedings of 10th World Congress on Automatic Control – IFAC*, Vol. 3, pp. 355-360, Munich, 1987.

[3] G. G. Parma, B. R. Menezes, A. P. Braga, J. C. R. Oliveira e L. A. Aguirre, "Observador neural de fluxo estatórico com treinamento *on-line*," *in Proceedings of XII Brazilian Automatic Control Conference – XII CBB*, vol. IV, pp. 1301-1306, 1998.

[4] J-P. Martens and N. Weymaere, "An equalized error backpropagation algorithm for the on-line training of multilayer perceptrons," *IEEE Trans.on Neural Networks*, vol. 13, No. 3, pp. 532-541, May 2002.

[5] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics 5*, pp. 115-133, 1943.

[6] D. E. Rumelhart, G. E. Hinton and R. J. Willians, *Learning internal representations by error propagation*.. Parallel Distributed Processing: Explorations in the microstructure of cognition, vol. I: Foundations, pp. 318-362, Cambridge, MA: MIT Press, 1986.

[7] A. Sabanovic, K. Jezernik and M. Rodic, "Neural network application in sliding mode control systems," *in IEEE Workshop on variable structure systems*, 1996.

[8] V. I. Utkin, *Sliding regimes an their applications in variable-structure systems*, Moskow, Nauka, 1974.