We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

28

Sliding Mode Control Approach for Training On-line Neural Networks with Adaptive Learning Rate

Ademir Nied and José de Oliveira
Department of Electrical Engineering, State University of Santa Catarina
Brazil

1. Introduction

This chapter includes contributions to the theory of on-line training of artificial neural networks (ANN), considering the multilayer perceptrons (MLP) topology. By on-line training, we mean that the learning process is conducted while the signal processing is being executed by the system, i.e., the neural network continuously adjusts its free parameters from the variations in the incident signal in real time (Haykin, 1999).

An artificial neural network is a massively parallel distributed processor made up of simple processing units, which have a natural tendency to store experimental knowledge and make it available for use (Haykin, 1999). These units (also called neurons) are non-linear adaptable devices, although very simple in terms of computing power and memory. However, when linked, they have enormous potential for nonlinear mappings. The learning algorithm is the procedure used to do the learning process, whose function is to modify the synaptic weights of the network in an orderly manner to achieve a desired goal of the project (Haykin, 1999).

Although initially used only in problems of pattern recognition and signal processing and image, today, the ANN are used to solve various problems in several areas of human knowledge.

An important feature of ANN is its ability to generalize, i.e., the ability of the network to provide answers in relation to standards unknown or not presented during the training phase. Among the factors that influence the generalization ability of ANN, we cite the network topology and the type of algorithm used to train the network.

The network topology refers to the number of inputs, outputs, number of layers, number of neurons per layer and activation function. From the work of Cybenko (1989), networks with the MLP topology had widespread use, because they possessed the characteristic of universal approximator of continuous functions. Basically, an MLP network is subdivided into the following layers: input layer, intermediate or hidden layer(s) and output layer. The operation of an MLP network is synchronous, i.e., given an input vector, it is propagated to the output by multiplying by the weights of each layer, applying the activation function (the model of each neuron of the network includes a non-linear activation function, being the non-linearity differentiable at any point) and propagating this value to the next layer until the output layer is reached.

Issues such as flexibility of the system to avoid biased solutions (*under□tting*) and, conversely, limiting the complexity of network topology, thus avoiding the variability of solutions

(*over□tting*), are inherent aspects to define the best topology for an MLP. This balance between bias and variance is known in the literature as "the dilemma between bias and variance" (German et al., 1992).

Several algorithms that seek to improve the generalization ability of MLP networks are proposed in the literature (Reed, 1993). Some algorithms use construction techniques, changing the network topology. That is, from a super-sized network already trained, methods of *pruning* are applied in order to determine the best topology considering the best balance between bias and variance. Other methods use restriction techniques of the weights values of MLP networks without changing the original topology. However, it is not always possible to measure the complexity of a problem, which makes the choice of network topology an empirical process.

Regarding the type of algorithm used for training MLP networks, the formulation of the backpropagation algorithm (BP) (Rumelhart et al., 1986) enabled the training of fedforward neural networks (FNN). The algorithm is based on the BP learning rule for error correction and can be viewed as a generalization of the least mean square algorithm (LMS) (Widrow & Hoff, 1960), also known as delta rule.

However, because the BP algorithm presents a slow convergence, dependent on initial conditions, and being able to stop the training process in regions of local minima where the gradients are zero, other methods of training appeared to correct or minimize these deficiencies, such as Momentum (Rumelhart et al., 1986), QuickProp (Fahlman, 1988), Rprop (Riedmiller & Braun, 1993), setting the learning rate (Silva & Almeida, 1990; Tollenaere, 1990), the conjugate gradient algorithm (Brent, 1991), the Levenberg-Marquardt algorithm (Hagan & Menhaj, 1994; Parisi et al., 1996), the fast learning algorithm based on the gradient descent in the space of neurons (Zhou & Si, 1998), the learning algorithm in real-time neural networks with exponential rate of convergence (Zhao, 1996), and recently a generalization of the BP algorithm, showing that the most common algorithms based on the BP algorithm are special cases of the presented algorithm (Yu et al., 2002).

However, despite the previously mentioned methods accelerating the convergence of network training, they cannot avoid areas of local minima (Yu et al., 2002), i.e., regions where the gradients are zero because the derivative of the activation function has a value of zero or near zero, even if the difference between the desired output and actual output of the neuron is different from zero.

Besides the problems mentioned above, it can be verified that the learning strategy of training algorithms based on the principle of backpropagation is not protected against external disturbances associated with excitation signals (Efe & Kaynak, 2000; 2001).

The high performance of variable structure system control (Itkis, 1976) in dealing with uncertainties and imprecision have motivated the use of the sliding mode control (SMC) (Utkin, 1978) in training ANN (Parma et al., 1998a). This approach was chosen for three reasons: because it is a well established theory, it allows for the adjustment of parameters (weights) of the network, and it allows an analytical study of the gains involved in training. Thus, the problem of the training of MLP networks is treated and solved as a problem of control, inheriting characteristics of robustness and convergence inherent in systems that use SMC.

The results presented in Efe & Kaynak (2000), Efe et al. (2000) have shown that the convergence properties of gradient-based training strategies widely used in ANN can be improved by utilizing the SMC approach. However, the method presented indirectly uses the Variable Structure Systems (VSS) theory. Some studies on the direct use of SMC strategy

are also reported in the literature. In Sira-Ramirez & Colina-Morles (1995) the zero-level set of the learning error variable in Adaline neural networks is regarded as a sliding surface in the space of learning parameters. A sliding mode trajectory can then be induced, in finite time, on such a desired sliding manifold. The proposed method was further extended in Yu et al. (1998) by introducing adaptive uncertainty bound dynamics of signals. In Topalov et al. (2003), Topalov & Kaynak (2003) the sliding mode strategy for the learning of analog Adaline networks, proposed by Sira-Ramirez & Colina-Morles (1995), was extended to a more general class of multilayer networks with a scalar output.

The first SMC learning algorithm for training multilayer perceptron (MLP) networks was proposed by Parma et al. (1998a). Besides the speed up achieved with the proposed algorithm, control theory is actually used to guide neural network learning as a system to be controlled. It also differs from the algorithms in Sira-Ramirez & Colina-Morles (1995), Yu et al. (1998) and Topalov et al. (2003), due to the use of separate sliding surfaces for each network layer. A comprehensive review of VSS and SMC can be seen in Hung et al. (1993), and a survey about the fusion of computationally intelligent methodologies and SMC can be found in Kaynak et al. (2001).

Although the methodology used by Parma et al. (1998a) makes it possible to determine the limits of parameters involved in the training of MLP networks, their complexity still makes it necessary to use heuristic methods to determine the most appropriate gain to be used in order to ensure the best network performance for a particular training.

In this chapter, an algorithm for on-line ANN training based on SMC is presented. The main feature of this procedure is the adaptability of the gain (learning rate), determined iteratively for every weight update, and obtained from only one sliding surface.

To evaluate the algorithm, simulations were performed considering two distinct applications: function approximation and a neural-based stator flux observer of an induction motor (IM). The network topology was defined according to the best possible response with the fewest number of neurons in the hidden layer without compromising the ability of network generalization. The network used in the simulations has only one hidden layer, differing in the number of neurons in this layer and the number of inputs and outputs of the network, which were chosen according to the application for the MLP.

2. The On-line adaptive MLP training algorithm

This section presents the algorithm with adaptive gain for on-line training MLP networks with multiple outputs that operates in quasi-sliding modes. The term "quasi-sliding regime" was introduced by Miloslavjevic (1985) to express the fact that the extension to the case of discrete time under the usual time for the continuous existence of a sliding regime, does not necessarily guarantee chattering around the sliding surface in the same way that it occurs in continuous time systems. Moreover, in Sarpturk et al. (1987) it was shown that the condition proposed by Miloslavjevic (1985) for the existence of a quasi-sliding mode could cause the system to become unstable. Now, let us specify how the quasi-sliding mode and the reaching condition are understood in this paper.

Definition 1. *Let us define a quasi-sliding mode in the ε vicinity of a sliding hyperplane $s(n) = 0$ for a motion of the system such that*

$$|s(n)| \leq \varepsilon \tag{1}$$

where the positive constant ε is called the quasi-sliding-mode band width (Bartoszewicz, 1998).

This definition is different from the one proposed in Gao et al. (1995) since it does not require the system state to cross the sliding plane $s(n) = 0$ in each successive control step.
The convergence of the system state to the sliding surface can be analyzed considering the convergence of the series

$$\sum_{n=1}^{\infty} s(n). \tag{2}$$

If the convergence of the series is guaranteed, then the system state will converge, at least assimptotically, to the sliding surface $s(n) = 0$.
Consider Cauchy's convergence principle (Kreyszig, 1993): The series $s_1 + s_2 + \cdots + s_n$ converges if and only if, for a given value $\varepsilon \in \Re^+$, a value N can be found such that $\mid s_{n+1} + s_{n+2} + \cdots + s_{n+p} \mid < \varepsilon$ for all $n > N$ e $p = 1, 2, \cdots$. A series is absolutely convergent if:

$$\sum_{n=1}^{\infty} |s(n)| \tag{3}$$

is convergent. To study the convergence of the series given by (3) the ratio test is used (Butkov, 1968). Thus, it holds that:

$$\left| \frac{s(n+1)}{s(n)} \right| \leq Q < 1. \tag{4}$$

Definition 2. *It is said that the system state converges to a quasi-sliding regime in the vicinity ε of a sliding surface $s(n) = 0$ if the following condition is satisfied:*

$$|s(n+1)| < |s(n)|. \tag{5}$$

Remark: From Definition 2, crossing the plane $s(n) = 0$ is allowed but not required.

Theorem 1. *Let $s(n) : \Re^2 \to \Re$, the sliding surface defined by $s(n) = CX1(n) + X2(n)$, where $\{C, X1(n)\} \in \Re^+$ and $X2(n) \in \Re$. If $X1(n) = E(n)$, being $E(n) = \frac{1}{2}\sum_{k=1}^{m_L} e_k^2(n)$ defined as the instantaneous value of the total energy of the error of all the neurons of the output layer of an MLP, where $e_k(n) = d_k(n) - y_k(n)$ is the error signal between the desired value and actual value at the output of the neuron k of the network output at iteration n, m_L is the number of neurons in the output layer of the network, and $X2(n) = \frac{X1(n) - X1(n-1)}{T}$ is defined as the variation of $X1(n)$ in a sample period of T, then, for the current state $s(n)$ to converge to a vicinity ε of $s(n) = 0$, it is necessary and sufficient that the network meet the following:*

$$sign(s(n))\,[C(X1(n+1) - X1(n)) + X2(n+1) - X2(n)] < 0 \tag{6}$$

$$sign(s(n))\,[C(X1(n+1) + X1(n)) + X2(n+1) + X2(n)] > 0, \tag{7}$$

being $sign(s(n)) = \begin{cases} +1, & s(n) \geq 0 \\ -1, & s(n) < 0 \end{cases}$ the sign function of $s(n)$. ◊

Proof: Defining the absolute value of the sliding surface as follows

$$|s(n)| = sign(s(n))s(n), \tag{8}$$

then, from (5) it holds that

$$|s(n+1)| < |s(n)| \Rightarrow sign(s(n+1))s(n+1) < sign(s(n))s(n).$$

As $sign(s(n))sign(s(n)) = 1$, yields

$$sign(s(n))[sign(s(n))sign(s(n+1))s(n+1) - s(n)] < 0.$$

If $sign(s(n+1)) = sign(s(n))$, then $sign(s(n))[s(n+1) - s(n)] < 0$. Replacing the definition of $s(n)$ as given by Theorem 1 yields

$$sign(s(n))\,[CX1(n+1) + X2(n+1) - (CX1(n) + X2(n))] < 0 \Rightarrow (6).$$

If $sign(s(n+1)) = -sign(s(n))$, then $sign(s(n))[-s(n+1) - s(n)] < 0$. Replacing the definition of $s(n)$ as given by Theorem 1 yields

$$sign(s(n))\,[CX1(n+1) + X2(n+1) + CX1(n) + X2(n)] > 0 \Rightarrow (7).$$

To prove that the conditions of Theorem 1 are sufficient, two situations must be established:

- The sliding surface is not crossed during convergence. In this situation, it holds that

  $$sign(s(n+1)) = sign(s(n)).$$

  Considering $s(n) = CX1(n) + X2(n)$ and $s(n+1) = CX1(n+1) + X2(n+1)$, one can write (6) as

  $$sign(s(n))[s(n+1) - s(n)] < 0 \Rightarrow sign(s(n+1))s(n+1) < sign(s(n))s(n),$$

  and using (8) yields $|s(n+1)| < |s(n)|$. The validity of (7) for this situation is trivial, i.e.:

  $$sign(s(n))[s(n+1) + s(n)] = |s(n+1)| + |s(n)| \Rightarrow (7).$$

- The sliding surface is crossed during convergence. Now, for this situation it holds that

  $$sign(s(n+1)) = -sign(s(n)).$$

  Considering, again, $s(n) = CX1(n) + X2(n)$ and $s(n+1) = CX1(n+1) + X2(n+1)$, one can write (7) as

  $$sign(s(n))[s(n+1) + s(n)] > 0 \Rightarrow sign(s(n+1))s(n+1) < sign(s(n))s(n),$$

  and using (8) yields $|s(n+1)| < |s(n)|$. The validity of (6) for this situation is trivial too, i.e.:

  $$sign(s(n))[s(n+1) - s(n)] = -|s(n+1)| - |s(n)| \Rightarrow (6).$$

□

From Theorem 1, it can be verified that (6) is responsible for the existence of a quasi-sliding regime for $s(n) = 0$, while (7) ensures the convergence of the network state trajectories to a vicinity of the sliding surface $s(n) = 0$. One can also observe that the reference term from the sliding surface signal $sign(s(n))$ determines the external and internal limits of the range of convergence for the following expressions:

$$C(X1(n+1) - X1(n)) + X2(n+1) - X2(n) \tag{9}$$

$$C(X1(n+1) + X1(n)) + X2(n+1) + X2(n). \tag{10}$$

To study the convergence of the sliding surface $s(n) = CX1(n) + X2(n)$, the decomposition of (9) and (10), with respect to a gain η, is necessary in order to obtain a set of equations for these variables and, from the conditions defined by Theorem 1, to determine an interval in $\Re$ due to a gain η, that can guarantee the convergence of the proposed method.

Theorem 2. *Let $s(n) : \Re^2 \rightarrow \Re$, the sliding surface defined by $s(n) = CX1(n) + X2(n)$, where $\{C, X1(n)\} \in \Re^+$ and $X2(n) \in \Re$. If $X1(n)$, $X2(n)$ and T are defined as in Theorem 1, then, for the current state $s(n)$ to converge to a vicinity ε of $s(n) = 0$, it is necessary and sufficient that the network meets the following:*

$$sign(s(n)) \left[c_1\eta^2 + c_2\eta - s(n) + CX1(n) \right] < 0 \tag{11}$$

$$sign(s(n)) \left[c_1\eta^2 + c_2\eta + s(n) + CX1(n) \right] > 0, \tag{12}$$

where $\{c_1, c_2\} \in \Re$. If the following restrictions are taken into account:

$$c_1 > 0 \tag{13}$$

$$c_2 < 0 \tag{14}$$

$$\Delta = c_2^2 - 4c_1c_3 > 0, \tag{15}$$

being $c_3 = \begin{cases} -s(n) + CX1(n), \text{ (in (11))} \\ s(n) + CX1(n), \text{ (in (12))} \end{cases}$ then, the existence of a limited region for the gain η that satisfies both conditions for convergence is guaranteed. ◊

Proof: Initially, consider that:

$$X1(n) = \frac{1}{2}\sum_{k=1}^{m_L}(d_k(n) - y_k(n))^2 = \frac{1}{2}\sum_{k=1}^{m_L}(d_k^2(n) - 2d_k(n)y_k(n) + y_k^2(n)), \tag{16}$$

$$X1(n+1) = \frac{1}{2}\sum_{k=1}^{m_L}(d_k^2(n+1) - 2d_k(n+1)y_k(n+1) + y_k^2(n+1)), \tag{17}$$

$$X1(n-1) = \frac{1}{2}\sum_{k=1}^{m_L}(d_k^2(n-1) - 2d_k(n-1)y_k(n-1) + y_k^2(n-1)), \tag{18}$$

$$X2(n+1) = \frac{X1(n+1) - X1(n)}{T}. \tag{19}$$

From (16), (17), (18), (19) and considering the definition of $X2(n)$ given by Theorem 1, one can derive the terms of (9) taking into account that $d_k(n-1) = d_k(n) = d_k(n+1) = d_k$. Thus, it holds:

$$\begin{aligned}
&C(X1(n+1) - X1(n)) + X2(n+1) - X2(n) = \\
&C(X1(n+1) - X1(n)) + \left(\frac{X1(n+1) - X1(n)}{T}\right) - \left(\frac{X1(n) - X1(n-1)}{T}\right) \\
&= \frac{1}{T}\left[(TC+1)X1(n+1) - (TC+2)X1(n) + X1(n-1)\right] \\
&= \frac{1}{T}\left[(TC+1)\frac{1}{2}\sum_{k=1}^{m_L}(d_k^2 - 2d_ky_k(n+1) + y_k^2(n+1))\right. \\
&\qquad \left. -(TC+2)\frac{1}{2}\sum_{k=1}^{m_L}(d_k^2 - 2d_ky_k(n) + y_k^2(n)) + \frac{1}{2}\sum_{k=1}^{m_L}(d_k^2 - 2d_ky_k(n-1) + y_k^2(n-1))\right] \\
&= \frac{1}{T}\frac{1}{2}\sum_{k=1}^{m_L}\left[TC(-2d_ky_k(n+1) + y_k^2(n+1) + 2d_ky_k(n) - y_k^2(n)) - 2d_ky_k(n+1)\right. \\
&\qquad \left. +y_k^2(n+1) + 4d_ky_k(n) - 2y_k^2(n) - 2d_ky_k(n-1) + y_k^2(n-1)\right].
\end{aligned} \tag{20}$$

In the same way, it is possible to derive the terms of (10) taking into account the same considerations used to derive (9). Thus:

$$
\begin{aligned}
&C(X1(n+1)+X1(n))+X2(n+1)+X2(n)= \\
&C(X1(n+1)+X1(n))+\left(\frac{X1(n+1)-X1(n)}{T}\right)+\left(\frac{X1(n)-X1(n-1)}{T}\right) \\
&=\frac{1}{T}\left[(TC+1)X1(n+1)+TCX1(n)-X1(n-1)\right] \\
&=\frac{1}{T}\left[(TC+1)\frac{1}{2}\sum_{k=1}^{m_L}(d_k^2-2d_ky_k(n+1)+y_k^2(n+1))\right. \\
&\qquad \left.+TC\frac{1}{2}\sum_{k=1}^{m_L}(d_k^2-2d_ky_k(n)+y_k^2(n))-\frac{1}{2}\sum_{k=1}^{m_L}(d_k^2-2d_ky_k(n-1)+y_k^2(n-1))\right] \\
&=\frac{1}{T}\frac{1}{2}\sum_{k=1}^{m_L}\left[TC(d_k^2-2d_ky_k(n+1)+y_k^2(n+1)+d_k^2-2d_ky_k(n)+y_k^2(n))\right. \\
&\qquad \left.-2d_ky_k(n+1)+y_k^2(n+1)-2d_ky_k(n-1)-y_k^2(n-1)\right]. \qquad (21)
\end{aligned}
$$

From (20) and (21) one can identify the term $y_k(n+1)$ as the target variable from which it is possible to obtain the gain η. Then, doing

$$y_k(n+1)=y_k(n)+c\eta, \qquad (22)$$

$$y_k^2(n+1)=y_k^2(n)+2y_k(n)c\eta+(c\eta)^2, \qquad (23)$$

replacing (22), (23) in (20), (21), respectively, and considering $e_k(n)=d_k-y_k(n)$, yields:

$$
\begin{aligned}
&\frac{1}{T}\frac{1}{2}\sum_{k=1}^{m_L}\left[(TC+1)c^2\eta^2-2(TC+1)ce_k(n)\eta\right. \\
&\qquad \left.+2d_ky_k(n)-y_k^2(n)-2d_ky_k(n-1)+y_k^2(n-1)\right] \qquad (24)
\end{aligned}
$$

and

$$
\begin{aligned}
&\frac{1}{T}\frac{1}{2}\sum_{k=1}^{m_L}\left[(TC+1)c^2\eta^2-2(TC+1)ce_k(n)\eta\right. \\
&\qquad +2TC(d_k-y_k(n))^2-2d_ky_k(n) \\
&\qquad \left.+y_k^2(n)+2d_ky_k(n-1)-y_k^2(n-1)\right]. \qquad (25)
\end{aligned}
$$

Finally, taking into account the result of $X1(n)-X1(n-1)$, one can obtain the conditions (11) and (12) defined in the Theorem 2, with the coefficients given by:

$$c_1=\frac{1}{2}\left(C+\frac{1}{T}\right)c^2$$

$$c_2=-\left(C+\frac{1}{T}\right)\sum_{k=1}^{m_L}ce_k(n)$$

$$c_3 = \begin{cases} -s(n) + CX1(n), \text{ (in (11))} \\ s(n) + CX1(n), \text{ (in (12))}. \end{cases} \tag{26}$$

To analyze the intervals of convergence limited by the conditions of (11) and (12) it is necessary to determine the limits of these intervals. It can be verified that the intervals of convergence are obtained from a parabola, the concavity of this parabola being determined by the value of c_1 (in this case, positive concavity, since $c_1 > 0$).
The general form for the quadratic equation related to the convergence conditions can be written as:

$$c_1\eta^2 + c_2\eta + c_3 \tag{27}$$

where c_3 is the independent term. Considering the value of $\Delta = c_2^2 - 4c_1c_3$ and taking into account that $c_1 > 0$, the roots of (27) are given by:

$$\Delta = c_2^2 - 4|c_1|c_3. \tag{28}$$

According to (28), the value of Δ is related to the signal and the module of the sliding surface $s(n)$. From these considerations, one can proceed with the following analysis:

- If $s(n) > 0$:
  - **(a)** $c_1\eta^2 + c_2\eta - s(n) + CX1(n) < 0$
    - **(1)** $|s(n)| > CX1(n) \Rightarrow c_3 < 0$.
      Roots: $\Delta = c_2^2 + 4|c_1||c_3| \Rightarrow \Delta > c_2^2$. Considering $\Delta = c_2^2\xi_1^2$, being $\xi_1 > 1$, the roots can be written as:
      $$\eta = -\frac{c_2}{2c_1} \pm \left|\frac{c_2\xi_1}{2c_1}\right| \tag{29}$$
    - **(2)** $|s(n)| < CX1(n) \Rightarrow c_3 > 0$
      Roots: $\Delta = c_2^2 - 4|c_1||c_3| \Rightarrow \Delta < c_2^2$. There are two possible variations for Δ:
      $1^a)$ $0 < \Delta < c_2^2$: Considering $\Delta = \frac{c_2^2}{\xi_1^2}$, the roots can be written as:
      $$\eta = -\frac{c_2}{2c_1} \pm \left|\frac{c_2}{2c_1\xi_1}\right| \tag{30}$$
      $2^a)$ $\Delta \leq 0$: This condition is not considered because it does not meet the restriction (15).
  - **(b)** $c_1\eta^2 + c_2\eta + s(n) + CX1(n) > 0$ Roots: $\Delta = c_2^2 - 4|c_1||c_3| \Rightarrow \Delta < c_2^2$. There are two possible variations for Δ:
    $1^a)$ $0 < \Delta < c_2^2$: Considering $\Delta = \frac{c_2^2}{\xi_2^2}$, being $\xi_2 > \xi_1$, the roots can be written as:
    $$\eta = -\frac{c_2}{2c_1} \pm \left|\frac{c_2}{2c_1\xi_2}\right| \tag{31}$$
    $2^a)$ $\Delta \leq 0$: This condition is not considered because it does not meet the restriction (15).

From (29), (30) and (31) the following relationship can be established:

$$\left|\frac{c_2}{2c_1\xi_2}\right| < \left|\frac{c_2}{2c_1\xi_1}\right| < \left|\frac{c_2\xi_1}{2c_1}\right|. \tag{32}$$

Considering $(-\frac{c_2}{2c_1})$ as the center point of convergence intervals and observing (32), a diagram can be drawn identifying, in bold, the intervals of convergence for $s(n) > 0$ as shown in Figure 1.

- If $s(n) < 0$:

  (a) $c_1\eta^2 + c_2\eta - s(n) + CX1(n) > 0 \Rightarrow c_1\eta^2 + c_2\eta + s(n) + CX1(n) > 0$

  Roots: $\Delta = c_2^2 - 4|c_1||c_3| \Rightarrow \Delta < c_2^2$. There are two possible variations for Δ:

  $1^a)$ $0 < \Delta < c_2^2$: Considering $\Delta = \frac{c_2^2}{\xi_2^2}$, the roots can be written as:

$$\eta = -\frac{c_2}{2c_1} \pm \left|\frac{c_2}{2c_1\xi_2}\right| \tag{33}$$

  $2^a)$ $\Delta \leq 0$: This condition is not considered because it does not meet the restriction (15).

  (b) $c_1\eta^2 + c_2\eta + s(n) + CX1(n) < 0 \Rightarrow c_1\eta^2 + c_2\eta - s(n) + CX1(n) < 0$

  (1) $|s(n)| > CX1(n) \Rightarrow c_3 < 0$.

  Roots: $\Delta = c_2^2 + 4|c_1||c_3| \Rightarrow \Delta > c_2^2$. Considering $\Delta = c_2^2\xi_1^2$, the roots can be written as:

$$\eta = -\frac{c_2}{2c_1} \pm \left|\frac{c_2\xi_1}{2c_1}\right| \tag{34}$$

  (2) $|s(n)| < CX1(n) \Rightarrow c_3 > 0$

  Roots: $\Delta = c_2^2 - 4|c_1||c_3| \Rightarrow \Delta < c_2^2$. There are two possible variations for Δ:

  $1^a)$ $0 < \Delta < c_2^2$: Considering $\Delta = \frac{c_2^2}{\xi_1^2}$, the roots can be written as:

$$\eta = -\frac{c_2}{2c_1} \pm \left|\frac{c_2}{2c_1\xi_1}\right| \tag{35}$$

  $2^a)$ $\Delta \leq 0$: This condition is not considered because it does not meet the restriction (15).

From (33), (34) and (35), it can be established the same relationship defined in (32) and, therefore, the diagram can be drawn identifying, in bold, the intervals of convergence for $s(n) < 0$, as shown in Figure 1. □

Remark: The Theorem 2 guarantees the existence of real intervals for the gain η to satisfy the convergence conditions. However, the Theorem 2 does not guarantee, *directly*, the existence of a *positive interval* for the gain η. Both for $s(n) > 0$ and $s(n) < 0$, it is assured that at least one positive real root exists, which reinforces the existence of a positive interval for η. In (30), (31), (33) and (35), the existence of positive real roots is conditioned by $-\frac{c_2}{2c_1} > 0$. As $c_1 > 0$, the condition is: $-c_2 > 0 \Rightarrow c_2 < 0$, which can be easily verified from the application of the methodology developed in a two-layer MLP.

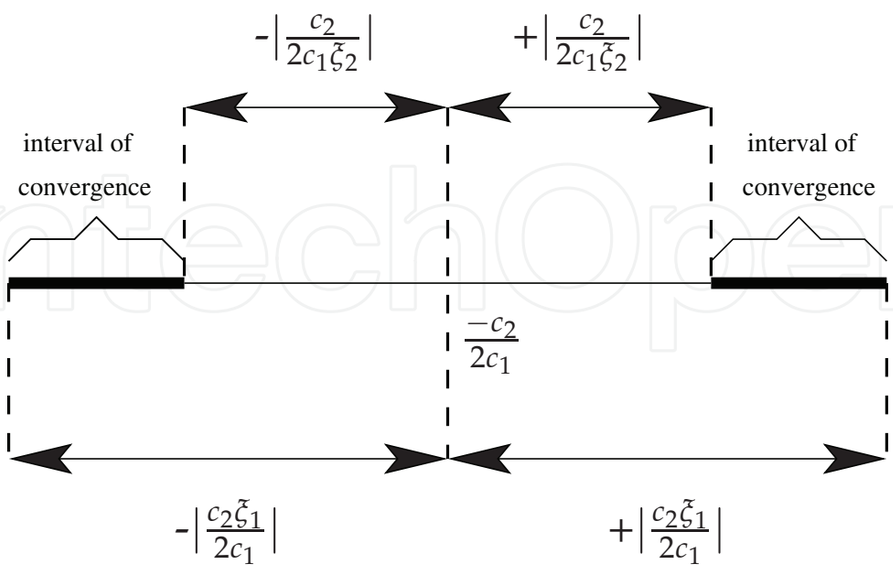


Fig. 1. Intervals of convergence for the algorithm with adaptive gain.

Once $s(n)$ is related to the network topology used, to verify the existence of a positive interval for the gain η, it is necessary to analyze the behavior of convergence conditions for the linear perceptron, the nonlinear perceptron and the two-layer MLP network with linear output. The choice of an MLP network topology was made in order to make the calculations involved in determining the network response to a stimulus simpler, yet still effective.

2.1 Determination of η for the linear perceptron

Let the output, at discrete-time n, of a neuron perceptron with linear activation function be given by:

$$y(n) = \sum_{j=1}^{m_0} w_j(n)x_j(n), \tag{36}$$

where m_0 is the number of inputs of the neuron. The analysis for the determination of the intervals for the gain η is performed for each input pattern of the neuron.
The output of the neuron at time $n+1$ is given by:

$$y(n+1) = y(n) + \Delta y(n) = y(n) + \sum_{j=1}^{m_0} \Delta w_j(n)x_j(n). \tag{37}$$

To calculate (37), it is necessary to determine $\Delta w_j(n)$, which represents the adjustment of the weights of the perceptron at time n. An immediate expression can be obtained from the Delta rule, which gives rise to the LMS algorithm or learning algorithm of gradient descent. Thus, it yields:

$$\Delta w_j(n) = -\eta \frac{\partial E(n)}{\partial w_j(n)} = -\eta\ 2\ \frac{1}{2}\ (d(n) - y(n))(-1)\frac{\partial y(n)}{\partial w_j(n)} = \eta e(n)x_j(n). \tag{38}$$

Once $\Delta w_j(n)$ is set, $y(n+1)$ can then be calculate as follows:

$$y(n+1) = y(n) + e(n) \sum_{j=1}^{m_0} x_j^2(n)\eta = y(n) + c\eta. \tag{39}$$

Therefore, using (39) and considering $c = e(n) \sum_{j=1}^{m_0} x_j^2(n)$, the expressions for the coefficients c_1, c_2 *e* c_3 of (26) can be obtained:

$$\begin{aligned} c_1 &= \frac{1}{2}\left(C + \frac{1}{T}\right) c^2 = \frac{1}{2}\left(C + \frac{1}{T}\right) e^2(n) \left(\sum_{j=1}^{m_0} x_j^2(n)\right)^2 \\ c_2 &= -\left(C + \frac{1}{T}\right) ce(n) = -\left(C + \frac{1}{T}\right) e^2(n) \sum_{j=1}^{m_0} x_j^2(n) \\ c_3 &= \begin{cases} -s(n) + CX1(n), \text{ (in (11))} \\ s(n) + CX1(n), \text{ (in (12))}. \end{cases} \end{aligned} \tag{40}$$

After determining the coefficients c_1, c_2 *e* c_3, the Theorem 2 can be applied to determine the intervals of convergence for the gain η.

2.2 Determination of η for the non-linear perceptron

The output characteristic of this type of neuron is given by:

$$y(n) = \varphi\left(\sum_{j=1}^{m_0} w_j(n) x_j(n)\right), \tag{41}$$

where $\varphi(\cdot)$ is the neuron activation function, continuous and differentiable.

The approach used to determine the neuron output is an approximation of the activation function through its decomposition into a Taylor series, instead of propagating the output signal of the neuron using the inverse of activation function. This approach was chosen because the first terms of the Taylor series provide a significant simplification and mathematical cost reduction for the definition of the intervals of convergence, yet limit the ability of approximating the function to regions close to the point of interest.

Let the output, at time n, of a neuron perceptron with non-linear activation function be given by (41). The output of the neuron at time $n+1$ can be written as:

$$y(n+1) = y(n) + \Delta y(n) = y(n) + \varphi\left(\sum_{j=1}^{m_0} \Delta w_j(n) x_j(n)\right). \tag{42}$$

Applying the decomposition of the first-order Taylor series in (42), yields:

$$y(n+1) = y(n) + \dot{y}(n) \sum_{j=1}^{m_0} \Delta w_j(n) x_j(n), \tag{43}$$

where $\left|\sum_{j=1}^{m_0} \Delta w_j(n) x_j(n)\right| \leq \xi$. Using (38) for the variation of weights at time n, it is possible to define an interval for the gain η related to the first-order Taylor series:

$$\eta \leq \frac{\xi}{\left|e(n) \sum_{j=1}^{m_0} x_j^2(n)\right|}. \tag{44}$$

It can be verified that (44) limits the interval of the gain η in accordance with the desired accuracy (ξ) for the approximation of the activation function of the neuron. Rewriting (43) it follows that:

$$\begin{aligned} y(n+1) &= y(n) + \dot{y}(n) e(n) \sum_{j=1}^{m_0} x_j^2(n) \eta \\ &= y(n) + c\eta. \end{aligned} \tag{45}$$

Therefore, using (45) and considering $c = \dot{y}(n) e(n) \sum_{j=1}^{m_0} x_j^2(n)$, the expressions for the coefficients c_1, c_2 *e* c_3 of (26) can be obtained:

$$\begin{aligned} c_1 &= \frac{1}{2}\left(C + \frac{1}{T}\right) c^2 = \frac{1}{2}\left(C + \frac{1}{T}\right) \dot{y}^2(n) e^2(n) \left(\sum_{j=1}^{m_0} x_j^2(n)\right)^2 \\ c_2 &= -\left(C + \frac{1}{T}\right) ce(n) = -\left(C + \frac{1}{T}\right) \dot{y}(n) e^2(n) \sum_{j=1}^{m_0} x_j^2(n) \\ c_3 &= \begin{cases} -s(n) + CX1(n), \text{ (in (11))} \\ s(n) + CX1(n), \text{ (in (12))}. \end{cases} \end{aligned} \tag{46}$$

After determining the coefficients c_1, c_2 *e* c_3, observing the limits imposed by the Taylor series decomposition, the Theorem 2 can be applied to determine the intervals of convergence for the gain η.

2.3 Determination of η for two-layer MLP network

Let the linear output of the k-th neuron of a two-layer MLP network related to an output vector $\mathbf{x}(n)$ be:

$$y2_k(n) = \sum_{j=1}^{m_1+1} w2_{kj}(n) y1_j(n) = \sum_{j=1}^{m_1+1} w2_{kj}(n) \varphi\left(\sum_{i=1}^{m_0} w1_{ji}(n) x_i(n)\right).$$

Due to the existence of two layers, one must do the study of the interval of convergence for the output layer and hidden layer separately. Thus, it follows:

- Output layer: Considering only the weights of the output layer as the parameters of interest, the output k at time n of an MLP network with linear output is given by:

$$y2_k(n) = \sum_{j=1}^{m_1+1} w2_{kj}(n) y1_j(n). \tag{47}$$

  Assuming that the adjustment of weights is performed initially only in the weights of the output layer, (47) can be compared to (36) for the linear perceptron. In this case, the inputs

of neuron k correspond to the output vector of neurons in the hidden layer (plus the bias term) after the activation function, $\mathbf{y1}(n)$, and the weights, for the vector $\mathbf{w2}_k(n)$. The coefficients c_1, c_2 *e* c_3 are obtained from the use of the equations for the linear neuron *by applying the analysis to the network with multiple outputs*. Thus, the coefficients of the quadratic equation associated with the convergence conditions are defined as:

$$\begin{aligned} c_1 &= \frac{1}{2}\left(C+\frac{1}{T}\right)\sum_{k=1}^{m_2}\left[e_k^2(n)\left(\sum_{j=1}^{m_1+1} y1_j^2(n)\right)^2\right] \\ c_2 &= -\left(C+\frac{1}{T}\right)\sum_{k=1}^{m_2}\left(e_k^2(n)\sum_{j=1}^{m_1+1} y1_j^2(n)\right) \\ c_3 &= \begin{cases} -s(n)+CX1(n), \text{ (in (11))} \\ s(n)+CX1(n), \text{ (in (12))}. \end{cases} \end{aligned} \tag{48}$$

- Hidden layer: Now, we consider the adjustment of the weights of the hidden layer, $\mathbf{W1}(n)$. For this, the weights of the output layer are kept constant. Therefore, the k-th neuron of the MLP network with two layers with linear output is given by:

$$y2_k(n) = \sum_{j=1}^{m_1+1} w2_{kj}(n)\varphi\left(\sum_{i=1}^{m_0} w1_{ji}(n)x_i(n)\right). \tag{49}$$

The output at time $n+1$ is given by:

$$y2_k(n+1) = y2_k(n) + \Delta y2_k(n) = y2_k(n) + \sum_{j=1}^{m_1+1} w2_{kj}(n)\varphi\left(\sum_{i=1}^{m_0} \Delta w1_{ji}(n)x_i(n)\right). \tag{50}$$

Applying in (50) the decomposition of the first order Taylor series, we obtain:

$$y2_k(n+1) = y2_k(n) + \dot{y2}_k(n)\sum_{j=1}^{m_1+1} w2_{kj}(n)\sum_{i=1}^{m_0}\Delta w1_{ji}(n)x_i(n), \tag{51}$$

where $\left|\sum_{i=1}^{m_0}\Delta w1_{ji}(n)x_i(n)\right| \leq \xi$. It is possible to use (38) for the variation of weights at time n. However, for the hidden layer, there is not a desired response specified for the neurons in this layer. Consequently, an error signal for a hidden neuron is determined recursively in terms of the error signals of all neurons for which the hidden neuron is directly connected, i. e., $\Delta w1_{ji}(n) = \eta\sum_{k=1}^{m_2} e_k(n)w2_{kj}(n)x_i(n)$. From the expression of $\Delta w1_{ji}(n)$ it is possible to define an interval for the gain η of the Taylor series decomposition:

$$\eta \leq \frac{\xi}{\left|\sum_{k=1}^{m_2} e_k(n)w2_{kj}(n)\sum_{i=1}^{m_0} x_i^2(n)\right|}. \tag{52}$$

Although (52) is assigned to a single neuron, the limit for the gain η must be defined in terms of the whole network, choosing the lower limit associated with a network of neurons. Decomposing (51) yields:

$$y2_k(n+1) = y2_k(n) + \dot{y2}_k(n)\sum_{j=1}^{m_1+1}\sum_{k=1}^{m_2} e_k(n)w2_{kj}^2(n)\sum_{i=1}^{m_0} x_i^2(n)\eta, \tag{53}$$

Therefore, using (53) and considering $c = \dot{y2}_k(n) \sum_{j=1}^{m_1+1} \sum_{k=1}^{m_2} e_k(n) w2_{kj}^2(n) \sum_{i=1}^{m_0} x_i^2(n)$, the coefficients c_1, c_2 *e* c_3 can be obtained as follows:

$$
\begin{aligned}
c_1 &= \frac{1}{2}\left(C + \frac{1}{T}\right) \sum_{k=1}^{m_2} \left[\dot{y2}_k^2(n) \sum_{j=1}^{m_1+1} \sum_{k=1}^{m_2} e_k^2(n) \left(w2_{kj}^2(n)\right)^2 \left(\sum_{i=1}^{m_0} x_i^2(n)\right)^2 \right] \\
c_2 &= -\left(C + \frac{1}{T}\right) \sum_{k=1}^{m_2} \left(\dot{y2}_k^2(n) \sum_{j=1}^{m_1+1} \sum_{k=1}^{m_2} e_k^2(n) w2_{kj}^2(n) \sum_{i=1}^{m_0} x_i^2(n) \right) \\
c_3 &= \begin{cases} -s(n) + CX1(n), \text{ (in (11))} \\ s(n) + CX1(n), \text{ (in (12))}. \end{cases}
\end{aligned} \tag{54}
$$

Thus, from the coefficients obtained in (48) and (54), the Theorem 2 can be apply, with the final interval for the gain η determined by the intersection of the intervals defined by convergence equations obtained for the hidden layer and the output layer, observing the limit imposed by the Taylor series decomposition. It should be noted also that, in (48) and (54), the coefficients c_1, c_2 e c_3 are dependent on C e T. This implies that, for the determination of C, the sampling period should be taken into account.

3. Simulation results

This section shows the results obtained from simulations of the algorithm presented in Section 2. The simulations are performed considering two distinct applications. In Section 3.1 the proposed algorithm is used in the approximation of a sine function. Then, in Section 3.2, the proposed algorithm is used for observation of the stator flux of the induction motor.

3.1 On-line function approximation

This section presents the simulation results of applying the proposed algorithm for the learning real-time function $f(t) = e^{(-\frac{1}{3})} sin(3t)$. The following parameters were considered for the simulations: integration step = 10μs; simulation time = 2s; sampling period = 250μs. The same simulations were also performed considering the standard BP algorithm (Rumelhart et al., 1986), the algorithm proposed by Topalov et al. (2003), and two algorithms for real-time training provided by (Parma et al., 1999a;b). For these algorithms, the training gains (learning rates) were chosen in order to obtain the best result, using the same initial conditions for each of the algorithms simulated.

The network topology used in the simulation of the algorithms was as follows: an input, 5 neurons in the hidden layer and one neuron in the output. The size of the hidden layer of the MLP was defined according to the best possible response with the fewest number of neurons. The hyperbolic tangent function was used as the activation function for the hidden layer neurons. This same function was also used as the activation function for the neuron of the output layer in the standard BP algorithm and on the two algorithms proposed by (Parma et al., 1999a;b). For the algorithm presented in this paper and that proposed by Topalov et al. (2003), the linear output for the neuron of the output layer was used.

The simulation results of the proposed algorithm are shown in Figure 2. For the confidence interval, $\xi = 1.5$ was used to approximate the hyperbolic tangent function using the first-order Taylor series.

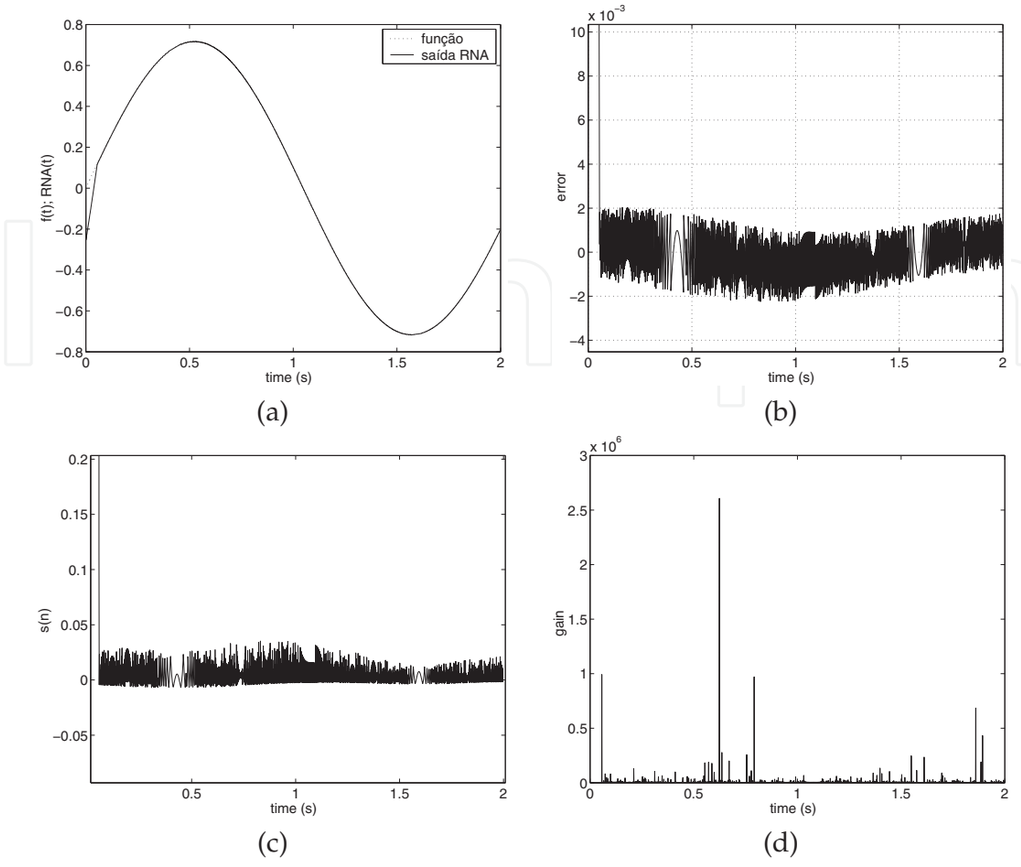


Fig. 2. Simulation results of the approximation of $f(t)$ using the presented algorithm: (a) output $f(t)$ x ANN(t); (b) error between output $f(t)$ and ANN output; (c) behavior of $s(n)$; (d) adaptive gain.

In the simulation, the value of 10,000 was adopted for the parameter C. The function $f(t)$ is shown dashed while the output of ANN is shown in continuous line. The graph of the approximation error for the sine function considered, the behavior of the sliding surface $s(n)$, and the training gains obtained from the proposed algorithm during the simulation time are also presented.

The fact that the proposed algorithm uses the gradient of error function with respect to weights, causes oscillations in the learning process, implying the need for high gains for the network training. These oscillations are also felt in the behavior of the sliding surface, as can be seen in the graph (c) of Figure 2.

Figure 3 shows the simulation results of the algorithms proposed by Parma et al. (1999a;b) and Topalov et al. (2003).

The coefficients and the gains of the algorithms were adjusted by obtaining the following values: 1st Parma algorithm - $C1$=$C2$=10000, $\eta 1$=3000, $\eta 2$=10; 2nd Parma algorithm - $C1$=$C2$=10000, $\eta 1$=200, $\eta 2$=100; Topalov algorithm - η=10. These three algorithms presented similar results, especially considering the time needed to reach the sine function, which is much smaller compared with the algorithm proposed in this paper. The proposed algorithm uses a gain adjustment which penalizes the reach time of the function $f(t)$. On the other

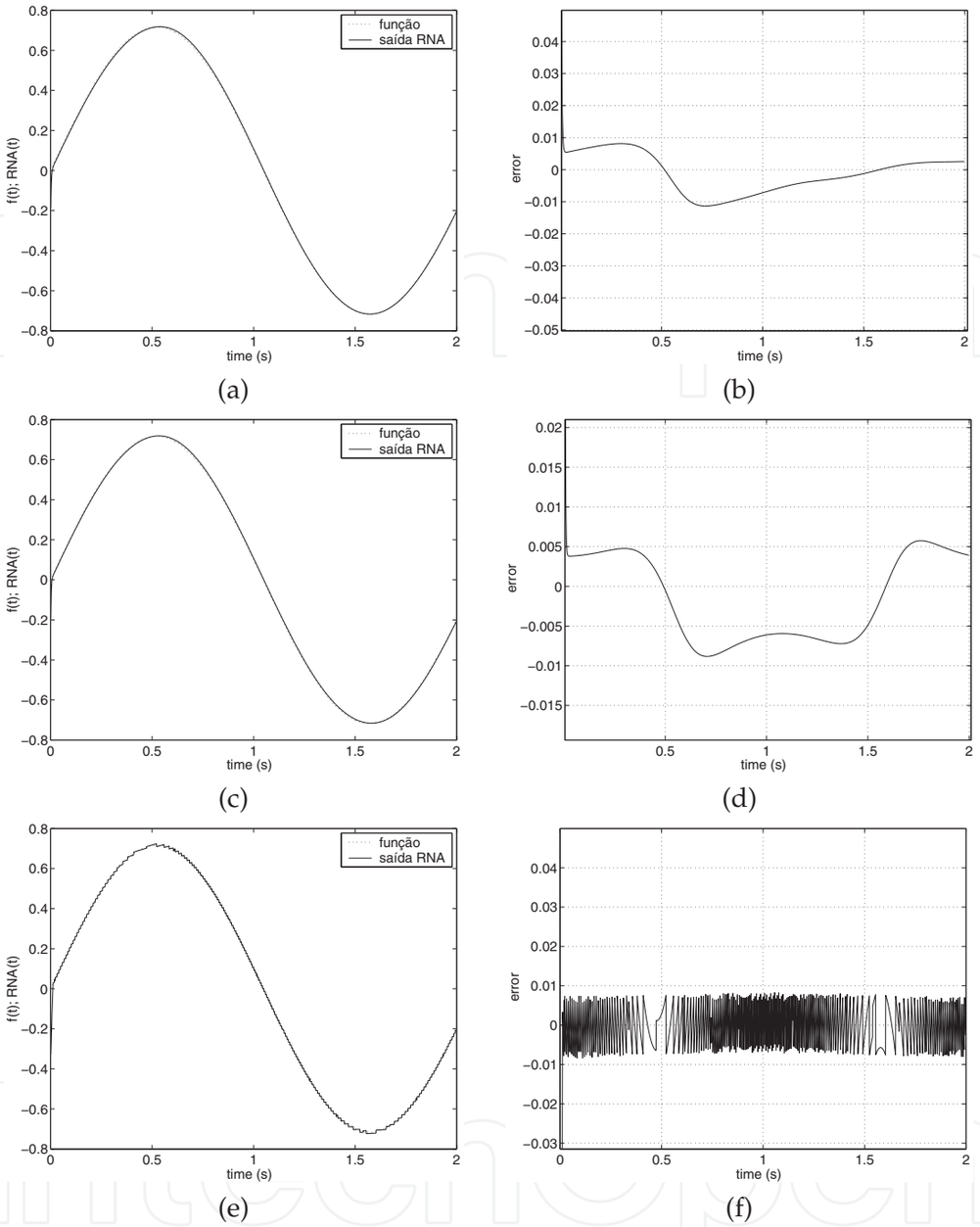
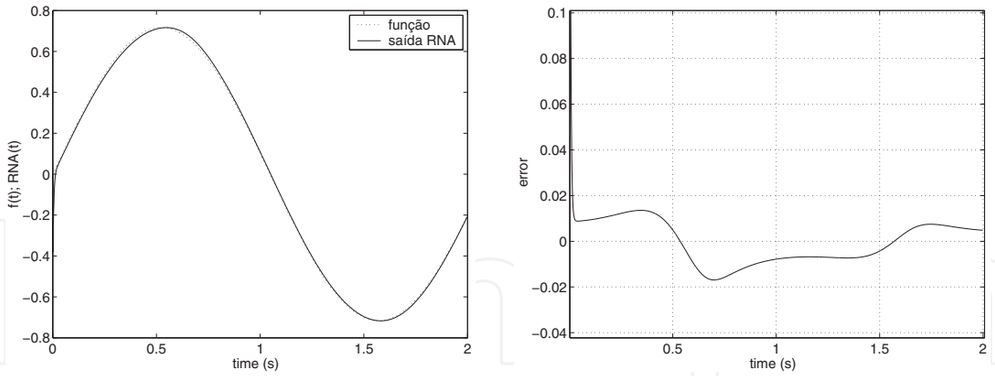


Fig. 3. Simulation results of the approximation of $f(t)$ using the proposed Parma and Topalov: graphs (a) and (b): - 1st Parma algorithm; graphs (c) e (d) - 2nd Parma algorithm; gráficos (e) e (f) - Topalov algorithm.

hand, if the errors of function approximation are compared, the proposed algorithm has better performance.

Finally, Figure 4 shows the results obtained using the standard BP algorithm. The adjusted values of gain for the hidden and output layers were, respectively, $\eta 1$=102 e $\eta 2$=12.

As can be easily verified, the standard algorithm BP had the highest error in the approximation of the considered function. This performance was expected for the various reasons outlined



(a) (b)

Fig. 4. Simulation results of the approximation of $f(t)$ using the standard BP algorithm: (a) output $f(t)$ x ANN(t); (b) error between output $f(t)$ and ANN output.

above. The results of this algorithm were presented as a reference, since this algorithm is the oldest of the simulated algorithms.

3.2 Induction motor stator flux neural Observer

Considering the IM drives, the correct estimation of the flux, either the stator, rotor and mutual, is the key to the successful implementation of any vector control strategy (Holtz & Quan, 2003).

The observation, in turn, is a closed loop estimation, which employs, in addition to the input signals, a feedback signal, obtained from the system output signals and the process model.

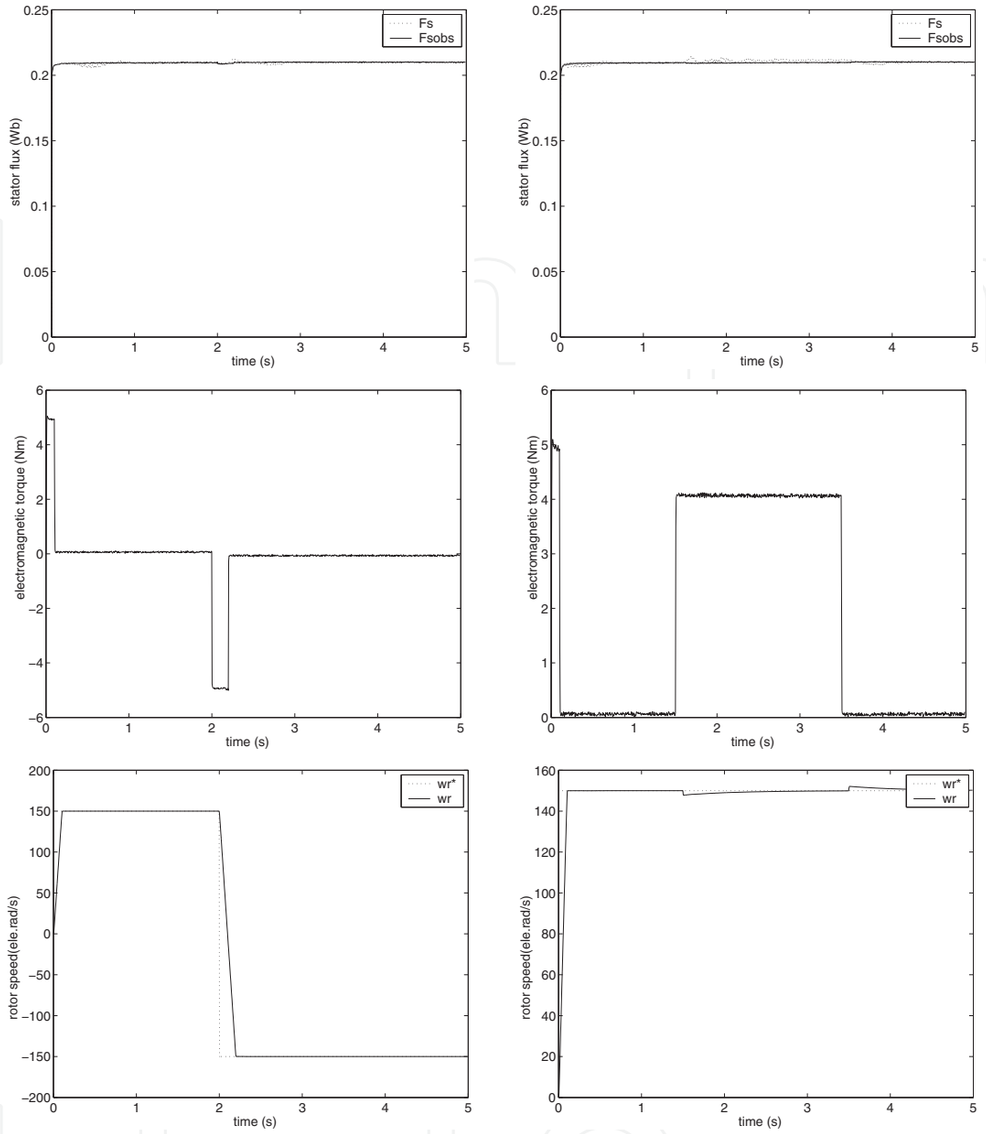
An important requirement for using an ANN for observing the motor flux, is that training should be done on-line. This approach allows a continuous adjustment of the network weights according to the requirements of the system in which the network operates, in this case, the IM. Figure 5 presents the simulation results of applying the proposed algorithm for training a neural network used as an IM stator flux observer. The following variables were considered: stator flux module (stator flux IM *versus* neural flux observer), electromagnetic torque and motor speed. The IM was submitted to the following transients: 1) start up and speed reversion with no load; 2) loading and unloading (constant torque) the motor at constant speed.

The IM flux can be estimated directly from the voltage equation given by (Novotny & Lipo, 1996):

$$\mathbf{v}_s = R_s \mathbf{i}_s + \frac{d\lambda_s}{dt} \Rightarrow \tag{55}$$

$$\lambda_s = \int (\mathbf{v}_s - R_s \mathbf{i}_s) dt. \tag{56}$$

The main reason for use of (56) is simplicity. The stator flux estimator is independent of the speed measurement if the stationary reference is adopted for the d-q axes (Kovács & Rácz, 1984). This fact makes the approach attractive for use in motor control without speed measurement. Moreover, one can see that the only parametric dependence is the stator resistance, which can be obtained with reasonable accuracy (Novotny & Lipo, 1996). Efficient solutions for the correction of off-set in the integrals of current and voltage can be verified in Holtz & Quan (2003).



(a) (b)

Fig. 5. Simulation results from neural observer: (a) speed reversal with no load in t=2s; (b) loading and unloading (constant torque) the motor at constant speed of 150 ele.rad/s in t=1.5s and t=3.5s, respectively.

Rewriting (56) considering d-q axes, it follows that:

$$\mathbf{v}_{sd} = R_s \mathbf{i}_{sd} + \frac{d\lambda_{sd}}{dt} \tag{57}$$

$$\mathbf{v}_{sq} = R_s \mathbf{i}_{sq} + \frac{d\lambda_{sq}}{dt}, \tag{58}$$

where R_s is the stator resistance; v_{sd} and v_{sq} are the d-q components of the stator voltage, i_{sd} and i_{sq} are the d-q components of the stator current, λ_{sd} and λ_{sq} are the d-q components of the stator flux, all of them in stator coordinates.

Thus, the d-q components of stator current are used as input of the ANN, and the d-q components of stator flux are the output of the network. The ANN used is the MLP 2-5-2, i.e., 2 inputs, 5 neurons in the hidden layer and 2 outputs. The number of neurons in the hidden layer was determined by analyzing the simulation results, aiming to reduce the computational cost without compromising the results generated by the network. Other studies using a neutral observer can be seen in Nied et al. (2003a) and Nied et al. (2003b).
The IM was submitted to the transients of start up and speed reversion with no load (Figure 5 (a)) and loading-unloading (constant torque) the motor at constant speed (Figure 5 (b)). Both transients are done under the motor speed condition of 150 elec.rad/s. The simulation time was 5 s. A good dynamic performance of the neural observer can be verified since the estimated stator flux tracks the stator reference flux, even during the transients applied to the motor.

4. Conclusion

Using the theory of sliding modes control, the problem of training MLP networks allows the analysis of the network as a system to be controlled, where the control variables are the weights, and the output of the network should follow the reference variable. From this, a methodology was used that allows us to obtain an adaptive gain, determined iteratively at each step of updating the weights, eliminating the need for using heuristics to determine the gain of the network. This methodology was used for on-line training of MLP networks with a linear activation function in the output layer.
The training of the ANN in real time requires a learning process to be performed while the signal processing is being executed by the system, resulting in the continual adjustment of free parameters of the neural network to variations in the incident signal in real time.
From the methodology, an algorithm was developed for on-line training of two-layer MLP networks with linear output. The algorithm presented is general, providing that there are one or more neurons in the output layer of the network.
Regarding the update of network weights, the algorithm updates the weights using the gradient of the error function with respect to the weights (BP algorithm). This weight correction law, despite being widely used for training MLP networks, has its weaknesses, such as the fact that the stability (not asymptotic stability) can only be guaranteed for a set of weights that corresponds to the overll minimum BP algorithm, according to Lyapunov stability theory.
By using the algorithm presented, it is possible to determine a resulting range for the gain η of the network, which is obtained through the intersection of the ranges defined for the hidden layer and output, noting the limit imposed by the Taylor series decomposition. However, the algorithm does not define the final value for the gain η. Thus, it is possible in principle, that any value within a range of positive results be used. Issues are not addressed by the optimization algorithm. However, bearing in mind the necessity of obtaining practical results from the application of the algorithm, we adopted a conservative solution using the gain value η obtained for the limit imposed by the Taylor series decomposition.
Due to the nature of the algorithm, applications that required adjustment of free parameters of the neural network in real time were selected for evaluation.
As a first application, the algorithm was used in the approximation of a sine function. The error of the approximation algorithm presented was the lowest compared with the values of the approximation error made by the other three algorithms simulated.

The other application was related to the use of the algorithm as an observer of the neural stator flux of IM. The results obtained show that the neural observer contributed to the good performance of the variables of flux, speed and torque.
From the simulation results of the algorithm, at least two features of this algorithm can be identify: 1) ease of use, since there is no necessity of determining the gain (or learning rate), which is obtained iteratively by the algorithm, 2) eliminates the need for any information regarding the mathematical model of the system in which the network operates.

5. References

Bartoszewicz, A. (1998). Discrete-time quasi-sliding-mode control strategies, *IEEE Tran. on Industrial Electronics* 45(4): 633–637.

Brent, R. P. (1991). Fast training algorithms for multilayer neural nets, *IEEE Trans. on Neural Networks* 2(3): 346–354.

Butkov, E. (1968). *Mathematical Physics*, Addison-Wesley.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function, *Mathematics of Control, Signals and Systems* 2(4): 304–314.

Efe, M. O. & Kaynak, O. (2000). Stabilizing and robustifying the error backpropagation method in neurocontrol applications, *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, IEEE Press, San Francisco, CA, pp. 1882–1887.

Efe, M. O. & Kaynak, O. (2001). Variable structure systems theory based training strategies for computationally intelligent systems, *Proceedings of the 27th Annual Conference of the IEEE Industrial Electronics Society*, IEEE Press, pp. 1563–1576.

Efe, M. O., Kaynak, O. & Wilamowski, B. M. (2000). Stable training of computationally intelligent systems by using variable structure systems technique, *IEEE Trans. on Industrial Electronics* 47(2): 487–496.

Fahlman, S. E. (1988). Faster-learning variations on backpropagation: an empirical study, *in* D. Touretzky, G. Hinton & T. Sejnowsky (eds), *Proceedings of the 1988 Connectionist Models Summer School*, Morgan Kaufmann, Sao Mateo, CA, pp. 38–51.

Gao, W., Wang, Y. & Homaifa, A. (1995). Discrete-time variable structure control systems, *IEEE Trans. on Industrial Electronics* 42(2): 117–122.

German, S., Bienenstock, E. & Dournsat, R. (1992). Neural networks and the bias/variance dilemma, *Neural Computation* 4(1): 1–58.

Hagan, M. T. & Menhaj, M. B. (1994). Training feedforword networks with the marquardt algorithm, *IEEE Trans. on Neural Networks* 5(6): 989–993.

Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation*, 2nd edn, Prentice-Hall, Englewood Cliffs, NJ.

Holtz, J. & Quan, J. (2003). Drift- and parameter-compensated flux estimator for persistent zero-stator-frequency operation of sensorless-controlled induction motors, *IEEE Trans. on Industry Applications* 39(4): 1052–1060.

Hung, J. Y., Gao, W. & Hung, J. C. (1993). Variable structure control: A survey, *IEEE Trans. on Industrial Electronics* 40(1): 2–22.

Itkis, U. (1976). *Control systems of variable structure*, John Wiley and Sons Inc., New York.

Kaynak, O., Erbatur, K. & Ertugrul, M. (2001). The fusion of computationally intelligent methodologies and sliding-mode control - a survey, *IEEE Trans. on Industrial Electronics* 48(1): 4–17.

Kovács, P. K. & Rácz, E. (1984). *Transient Phenomena in Electrical Machines*, Elsevier, Amsterdam, The Netherlands.

Kreyszig, E. (1993). *Advanced Engineering Mathematics*, 7th edn, John Wiley and Sons Inc.

Miloslavjevic, C. (1985). General conditions for the existence of a quasisliding mode on the switching hyperplane in discrete variable structure systems, *Automation and Remote Control* 46: 307–314.

Nied, A., Junior, S. I. S., Menezes, B. R., Parma, G. G. & Justino, J. C. G. (2003a). Comparative study on flux observers for induction motor drives, *Proceedings of the 7th Brazilien Power Electronics Conference*, Fortaleza, CE.

Nied, A., Junior, S. I. S., Parma, G. G. & Menezes, B. R. (2003b). On-line training algorithms for an induction motor stator flux neural observer, *Proceedings of 29th annual conference of the IEEE Industrial Electronics Society - IECON2003*, IEEE Press, Roanoke, VA, pp. 129–134.

Novotny, D. W. & Lipo, T. A. (1996). *Vector control and dynamics of AC drives*, 1st edn, Cleredon Press.

Parisi, R., Di Claudio, E. D., Orlandi, G. & Rao, B. D. (1996). A generalized learning paradigm exploiting the structure of feedforward neural networks, *IEEE Trans. on Neural Networks* 7(6): 1450–1460.

Parma, G. G., Menezes, B. R. & Braga, A. P. (1998a). Sliding mode algorithm for training multilayer neural network, *IEE Electronics Letters* 38(1): 97–98.

Parma, G. G., Menezes, B. R. & Braga, A. P. (1999a). Neural networks learning with sliding mode control: the sliding mode backpropagation algorithm, *International Journal of Neural Systems* 9(3): 187–193.

Parma, G. G., Menezes, B. R. & Braga, A. P. (1999b). Sliding mode backpropagation: control theory applied to neural networks learning, *Proceedings of the International Joint Conference on Neural Networks*, IEEE Computer Society Press, Washington-DC, pp. 1774–1778.

Reed, R. (1993). Pruning algorithms - a survey, *IEEE Trans. on Neural Networks* 4(5): 740–746.

Riedmiller, M. & Braun, H. (1993). A direct adaptive method for faster backpropagation learning: The rprop algorithm, *Proceedings of the Int. Conf. on Neural Networks*, San Francisco, CA, pp. 586–591.

Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1986). *Learning internal representation by error propagation*, Vol. 1 of *Parallel distribuited processing: explorations in the microstructure of cognition*, MIT Press, Cambridge, MA, pp. 318–362.

Sarpturk, S., Istefanopulos, Y. & Kaynak, O. (1987). On the stability of discrete-time sliding mode control systems, *IEEE Trans. on Automatic Control* 32(10): 930–932.

Silva, F. M. & Almeida, L. B. (1990). Speeding up backpropagation, *in* R. Eckmiller (ed.), *Advanced Neural Computers*, Amsterdam: Elsevier North Holland, pp. 151–158.

Sira-Ramirez, H. & Colina-Morles, E. (1995). A sliding mode strategy for adaptive learning in adalines, *IEEE Trans. on Circuits and Systems - I: Fundamental Theory and Applications* 42(12): 1001–1012.

Tollenaere, T. (1990). Supersab: Fast adaptive back propagation with good scaling properties, *Neural Networks* 3(5): 561–573.

Topalov, A. V. & Kaynak, O. (2003). A sliding mode strategy for adaptive learning in multilayer feedforward neural networks with a scalar output, *IEEE International Conference on Systems, Man and Cybernetics, 2003*, Vol. 2, IEEE Press, pp. 1636–1641.

Topalov, A. V., Kaynak, O. & Shakev, N. G. (2003). Variable structure systems approach for on-line learning in multilayer artificial neural networks, *Proc. IEEE 29th Annual Conference of the Industrial Electronics Society (2003)*, IEEE Press, Roanoke, VA, pp. 2989–2994.

Utkin, V. I. (1978). *Sliding modes and their application in Variable Structure Systems*, MIR, Moskow.

Widrow, B. & Hoff, M. E. (1960). Adaptive switching circuits, *IRE WESCON Convention Record*, Vol. 4, IRE, New York, pp. 96–104.

Yu, X., Efe, M. O. & Kaynak, O. (2002). A general backpropagation algorithm for feedforward neural networks learning, *IEEE Trans. on Neural Networks* 13(1): 251–254.

Yu, X., Zhihong, M. & Rahman, S. M. M. (1998). Adaptive sliding mode approach for learning in a feedforward neural networks, *Neural Computing and Applications* 7(4): 289–294.

Zhao, Y. (1996). On-line neural network learning algorithm with exponential convergence rate, *IEE Electronics Letters* 32(15): 1381–1382.

Zhou, G. & Si, J. (1998). Advanced neural network training algorithm with reduced complexity based on jacobian deficiency, *IEEE Trans. on Neural Networks* 9(3): 448–453.

Sliding Mode Control

Edited by Prof. Andrzej Bartoszewicz

ISBN 978-953-307-162-6
Hard cover, 544 pages
Publisher InTech
Published online 11, April, 2011
Published in print edition April, 2011

The main objective of this monograph is to present a broad range of well worked out, recent application studies as well as theoretical contributions in the field of sliding mode control system analysis and design. The contributions presented here include new theoretical developments as well as successful applications of variable structure controllers primarily in the field of power electronics, electric drives and motion steering systems. They enrich the current state of the art, and motivate and encourage new ideas and solutions in the sliding mode control area.

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen