Artificial Neural Networks Application to Boolean Input Systems Control

William Holderbaum, Régis Canart and Pierre Borne

Laboratoire d'Automatique et d' Informatique Industrielle de Lille
Ecole Centrale de Lille
BP 48 Cité Scientifique
59651 Villeneuve D'Ascq Cedex
FRANCE
e-mails : holderba@ec-lille.fr; p.borne@ec-lille.fr

Abstract: Boolean input systems have become important in the electrical industry. These systems mainly include power supply associated with converters and electric motors. In this paper we present a method for controlling Boolean input systems by using Artificial Neural Network. This method is based on classification of system variations associated with input configurations. The supervised backpropagation algorithm is used to train the networks. The training of the Artificial Neural Network and the control of Boolean input systems are presented.

The design procedure of control systems is implemented on a non-linear system. These results are applied to an electrical system composed of a synchronous motor and its power converter. The control of this system is performed on its speed.

Keywords: Boolean control, Commutation, State space, Classification, Artificial Neural Networks, Backpropagation.

William Holderbaum graduated in 1993 from the University of Reims, France and obtained the Master degree in Process Data Processing and Automatic Control. In 1994 he also obtained the DEA (equivalent qualifying exams for the Ph.D) in Instrumentation and Automatic Control from the University of Caen, France. He is currently a researcher at the Laboratory of Automatic Control and Informatics of Lille (LAIL), where he is achieving his Ph.D thesis. His scientific interests are Boolean control, motor control and neural networks for control application.

Régis Canart graduated in 1995 from the Engineering School ICAM in Lille, France. He also obtained the Ph.D degree in Automatic Control in 1997 from the University of Lille. He is now researcher in automatic control. His scientific interests are system identification, adaptive control and neural networks for control application.

Professor **Pierre Borne** graduated from the University of Lille in 1967 and obtained the Master degrees in Applied Mathematics, Physics, Mechanics and Electronics in 1968. In 1968 he also obtained the Diploma of Engineer of the IDN (French "Grand Ecole" with the new name "Ecole Centrale de Lille"). He obtained the Ph.D degree in Automatic Control in 1970 and Dsc. of Physics in 1976. He is now Professor of Automatic Control, head of the Automatic Control Department and Scientific Director of the Ecole Centrale de Lille.

He is involved in various international activities with IEEE/SMC Society and with IMACS. In 1996 he became a Fellow IEEE.

His research interests include stability analysis of non-linear systems, and various aspects of robust control of non-linear and complex systems.

1. Introduction

Recently there has been increasing interest in the development of efficient control strategies to improve dynamic behavior of power converters. The behavior of such systems is controlled by the switching ON and OFF of components such as thyristors or transistors. Among classical controllers which have been widely used there is the well-known P.W.M (Pulse Width Modulation) approach. This technique consists of controlling the process, using mean input values [1], [2], [3]. The regulation is often achieved by a P.I.D controller, showing good performances when disturbances are very small. On the contrary, when disturbances are large, the results of regulation become unsatisfactory. In practice, this method appears to be highly sensitive to the variation of system parameters.

Other techniques such as the classical Sliding Mode Control (SMC) [4], [5], [6] try to optimize the system response. This technique is characterized by discontinuous control actions on Variable Structure Systems (VSS) whose structure changes upon reaching a set of switching surfaces. The switching instants are determined by appropriate sliding surfaces (switching surfaces) which are chosen to achieve a desired dynamic response. Sliding Mode Control for multi-input systems is used to control electronic converters. The most distinctive feature of a Sliding Mode Control system is its robustness to parametric uncertainty and external disturbances. However it is very difficult to choose sliding surfaces in Multi Input Multi Output (MIMO) cases.

Artificial Neural Networks have been proved extremely useful in pattern recognition [7] [8] and control systems [8][9]. In this paper we propose an optimized neural network to control Boolean systems. At the beginning of the paper, a quick review pointing out the problem of

Boolean input systems is made. We then show how a multi-layer Neural Network is able to control the state of a switching circuit and to provide the control output which ensures that the trajectory is followed in the state space. This methodology consists of four parts. First, we determine the initialization of the Artificial Neural Network. Secondly data processing and filtering, which consist of reducing noises and disturbances, are discussed. The third part of this method deals with the neural network structure and the on-line training algorithm used to obtain a satisfactory network, which gives the correct control to apply on the system.

The proposed method has been implemented in simulation on a synchronous motor and its power converter in order to drive speed. Simulation results show a good response of the converter circuit with its load and confirm the validity of the neural approach.

2. Boolean Systems

Consider the system modeled by the state equation :

$$\dot{x} = f(x,u) \tag{1}$$

where $x = (x_1 \cdots\cdots x_n)^T \in \Re^n$ is the state vector, and $u = (u_1 \cdots\cdots u_m)^T \in \{0,1\}^m$ the input vector composed of Boolean variables.

The input vector u can take any configuration [10] among 2^m different vectors $Config_i(u)$ containing Boolean values such as :

$$\{Config_i(u) / i = 1 \cdots 2^m\} = \left\{ \begin{array}{ccccccccccc} \begin{pmatrix}0\\0\\0\\\vdots\\0\end{pmatrix} & \begin{pmatrix}1\\0\\0\\\vdots\\0\end{pmatrix} & \begin{pmatrix}0\\1\\0\\\vdots\\0\end{pmatrix} & \cdots & \begin{pmatrix}0\\0\\0\\\vdots\\1\end{pmatrix} & \begin{pmatrix}1\\1\\0\\\vdots\\0\end{pmatrix} & \begin{pmatrix}1\\0\\1\\\vdots\\0\end{pmatrix} & \cdots & \begin{pmatrix}1\\0\\0\\\vdots\\1\end{pmatrix} & \cdots & \begin{pmatrix}1\\1\\1\\\vdots\\1\end{pmatrix} \\ 1 & 2 & 3 & & m+1 & m+2 & m+3 & & i & & 2^m \end{array} \right\}$$

Between two commutations, the input u is a constant vector.

3. Boolean Control Using Artificial Neural Networks

The aim of the study is to apply Artificial Neural Networks to switching systems. To this end we use the principle formulated in [11][12]. We give a brief summary of this approach here. The goal is to determine the sequence of $Config_i(u)$ for vector u for which the state vector reaches a desired state, denoted as x_d.

Let us consider a vector ε representing a position error vector defined as follows :

$$\varepsilon = x_d - x_p \tag{2}$$

where x_p is the current position of the plant in the hyperplane associated with the state space and x_d is the target. We calculate vector $\{\vec{V}_i\}$ associated with $(\dot{x})_i$ for each configuration $Config_i(u)$ associated with the current position x_p :

$$(\dot{x})_i = f(x_p, Config_i(u)) \qquad i = 1,2,3,\cdots,2^m \tag{3}$$

When a configuration $Config_i(u)$ is applied to the system during a time t, the current state evolves in a particular direction. Among the direction set $\vec{V}_i$, we choose vector $\vec{V}_i$ such that the error position is reduced. Consequently the configuration $Config_i(u)$ can be deduced.

This procedure is illustrated by the example shown in Figure 1, where $x \in \Re^2$ and $u \in \{0,1\}^2$.

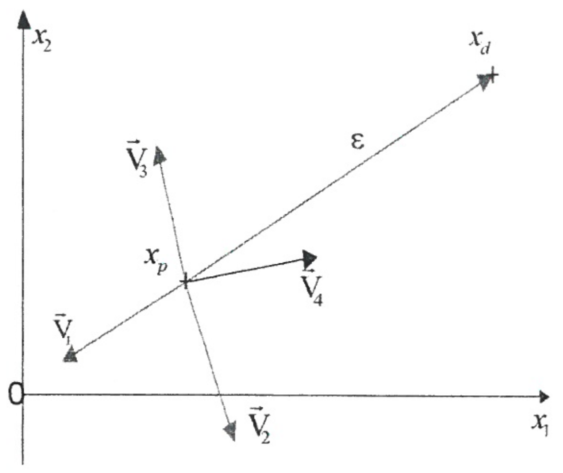

Figure 1. Control in the State Space

In that case, vector $\vec{V}_4$ is the best direction toward the desired state. $\vec{V}_4$ corresponds to the control $Config_4(u) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ applied to the process.

The disadvantage of this method is that of the knowledge of the model that is required to determine vector $\vec{V}_i$. We aim to solve this problem by using Artificial Neural Networks, exploiting their ability to control systems without any explicit knowledge of their models. The objective of the network is to find the configuration i such that the current state converges toward the desired state. This network is defined to establish the relationship between the control and the direction of the system evolution.

Structure of the Artificial Neural Network

Artificial Neural Networks can be defined as highly connected arrays of neurons [8]. The internal structure of a neuron is shown in Figure 2.

The internal activity of a single neuron computes the weighted sum of the inputs e_i (net) and passes this sum through a non-linear function f, according to :

$$\begin{cases} net = \sum_{i=1}^{n} w_i e_i + w_b \\ o = f(net) \end{cases} \qquad (4)$$

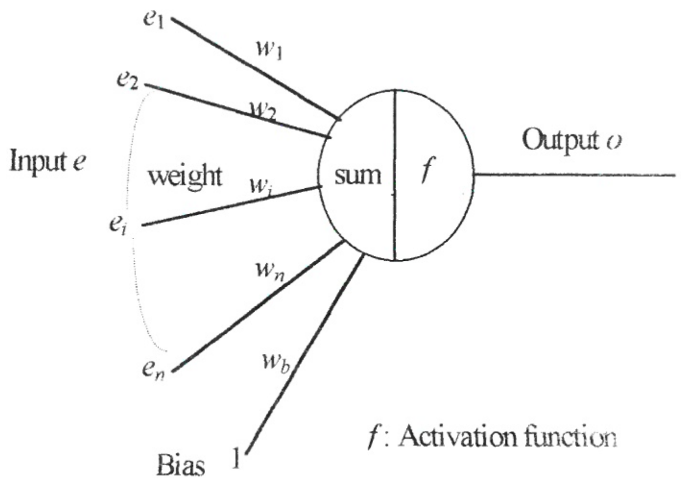

Figure 2. Neuron Model

Another term called the bias term w_b is associated with this sum. The function f used as a non-linear function is, for example, a sigmoid function given by :

$$f(net) = \frac{1}{1+e^{-net}} \qquad (5)$$

A layer is a set of elementary neurons. The neural networks used here are basically layers of neurons connected in cascade, with one input layer, one or more hidden layers and one output layer. The input layer is the sensory organ for the Artificial Neural Networks. Each neuron in a layer is connected to an adjacent neuron layer with different weights. Each neuron, except for the neurons of the input layer, receives signals from the neurons of the previous layer, weighted by the interconnect values between neurons. Consequently the output layer produces an output signal. The calculation of weights is performed with the learning algorithm, which is presented in the next Section.

To obtain the relationship between the different configurations and the state evolution, the network needs a minimum of three layers. The dimension of the input layer corresponds to the number of state variables to control. The output layer size is defined by the number of configurations such that each node is associated with one configuration. The choice of the number of hidden layer nodes is a compromise between efficiency and accuracy. Satisfactory results are obtainable if the number of hidden layer nodes is equal to the number of output nodes. The defined network achieves a classification of the 2^m configurations. The basic structure of a three-layer Artificial Neural Network capable to satisfactorily perform the Boolean control is shown in Figure 3.

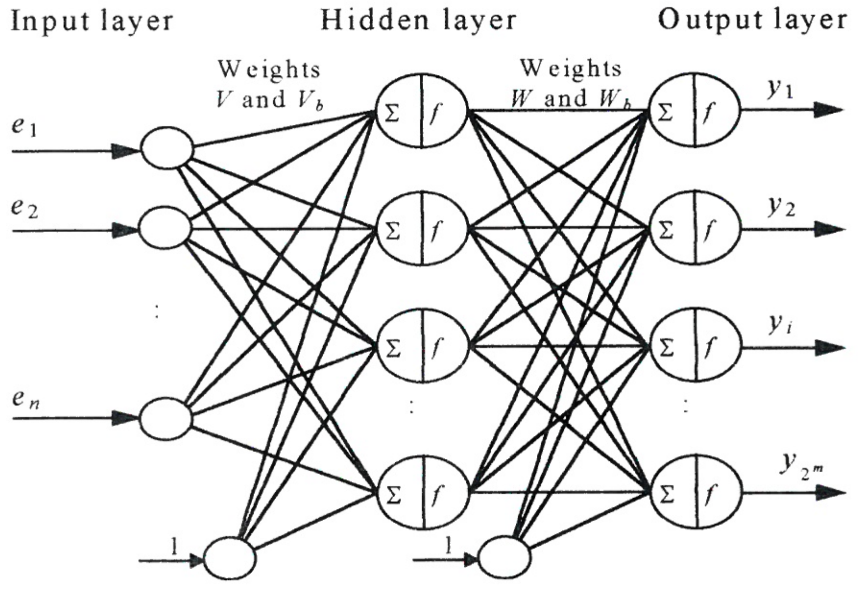

Figure 3. Scheme of An Artificial Neural Network

The propagation of the data is performed as follows. For the ith neuron of the output layer, the value y_i has the following shape :

$$y_i = f\left(\sum_{j=1}^{2^m}\left(w_{ij} f\left(\sum_{k=1}^{n} v_{jk} e_k + v_{b,j}\right) + w_{b,i}\right)\right) \tag{6}$$

where :

e_k : kth input of the network.

v_{jk} : the interconnection weights between the input and hidden layers.

$v_{b,j}$: the jth bias weight of the hidden layer.

w_{ij} : the weights between the hidden and output layers.

$w_{b,i}$: the ith bias weight of the output layer.

Clearly, the matrix form of the net output is

$$y = F\left(W\left(F\left(Ve + V_b\right)\right) + W_b\right) \tag{7}$$

with

$$F(NET) = \left[f(net_1) \quad f(net_2) \quad \cdots \quad f(net_{2^m})\right]^T$$

where net_z is the weighted sum of the z th neuron, and

$$NET = \left[net_1 \quad net_2 \quad \cdots \quad net_{2^m}\right]^T$$

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{2^m} \end{bmatrix}, \; e = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{bmatrix}, \; W = \left[w_{ij}\right],$$

$$V = \left[v_{jk}\right], W_b = \left[w_{b,i}\right] \text{ and}$$

$$V_b = \left[v_{b,j}\right].$$

Initialization

To initialize the network and the weights, two methods are proposed. In the case where the training has already been done once, the weights of the network are to be initialized as the latest used weights. Otherwise initial weights are chosen as small random numbers.

Secondly, the initialization step is necessary to acquire a set of measures to obtain initial training data which are defined in the next Section. This initialization is realized by applying each configuration $Config_i(u)$ for $i = \{1 \cdots 2^m\}$ to the process r times, in order to ensure that each configuration is used at least once. The acquisition of the measures is done at a sampling time $\mathrm{T_e}$. Consequently this initialization lasts $2^m r \mathrm{T_e}$. The network learning and plant control can only start after completing this preliminary step.

Data Processing

This Section presents the data processing which consists in defining suitable input-output training data of the neural network. Since the aim of the network is to provide the relationship between the state variations and the configurations applied to the plant, the input training data are defined as the state variations.

For each configuration i a state variation vector is determined as follows :

$$\left(\Delta x_p\right)_i(k) = \left(x_p\right)_i(k) - x_p(k-1) \qquad (8)$$

where $\left(\Delta x_p\right)_i(k)$ denotes the state variation at instant k when the configuration i is applied at the instant $k-1$.

However, this computation is sensitive to the measurement noise. To reduce the noise influence, a data filtering is necessary. The easiest and most efficient method is to perform for each configuration i the average on r variation vectors. For convenience, we denote $\left\{\Delta x_p\right\}_i$ as the set of the r latest state variation vectors of the configuration i, and the associated average as $\overline{\left\{\Delta x_p\right\}_i}$. This procedure is performed on-line. Hence, at each sampling time a new state vector variation $\left(\Delta x_p\right)_i(k)$ is acquired. The set $\left\{\Delta x_p\right\}_i$ and its average have then to be updated.

An illustration of this processing is provided below, in which we consider $x_p \in \Re^2$, $u \in \{0,1\}^2$ and $r = 3$.

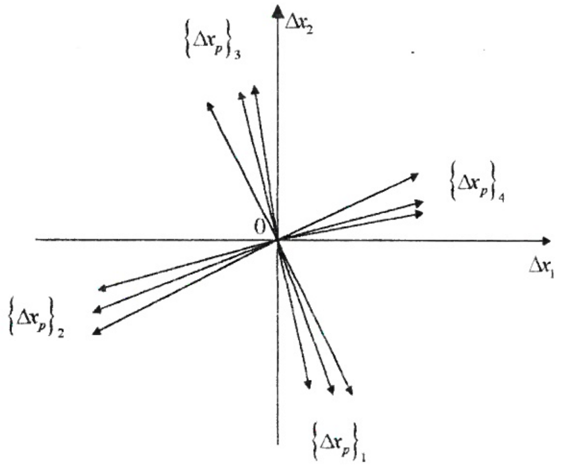

Figure 4. State Variation for Each Configuration $i = \{1 \cdots 4\}$

The results of the average $\overline{\left\{\Delta x_p\right\}_i}$ are the input data of the network. At this point, we have also to determine output data, which are used to train the neural network. By definition, the network is such that its ith output is associated with the configuration i. Therefore the desired output vector, also called target vector $\left(y^d\right)_i$ is chosen as follows :

$$\left(y^d\right)_i = \begin{bmatrix} y_1^d \\ y_2^d \\ \vdots \\ y_j^d \\ \vdots \\ y_{2^m}^d \end{bmatrix} \quad \text{with} \quad \begin{cases} y_j^d = 1 & \text{for } j = i, \\ y_j^d = 0 & \text{otherwise} \end{cases} \qquad (9)$$

Finally we deduce pairs of input-output data for each configuration i. For convenience, we

$$S = \left\{ \begin{matrix} \left(\overline{\left\{\Delta x_p\right\}_1}, \left(y^d\right)_1\right), \left(\overline{\left\{\Delta x_p\right\}_2}, \left(y^d\right)_2\right), \cdots, \\ \left(\overline{\left\{\Delta x_p\right\}_i}, \left(y^d\right)_i\right), \cdots, \left(\overline{\left\{\Delta x_p\right\}_{2^m}}, \left(y^d\right)_{2^m}\right) \end{matrix} \right\}$$

define a set S of samples given by :

Training Algorithm

The training problem consists to adjust online the weights using the set of data S. The learning strategy is based on the backpropagation algorithm [8]. The principle is to minimize for every input-output pair denoted $\left(e, y^d\right)$ of the set S, the quadratic criterion J defined as :

$$J = \frac{1}{2}\varepsilon_{nn}^T \varepsilon_{nn}$$

where the error vector ε_{nn} is given by the difference between the desired output y^d and the neural network output y obtained for the input e.

$$\varepsilon_{nn} = y^d - y$$

The algorithm used to minimize this criterion is based on the well-known gradient descent method, which gives for a weight w the following adaptation law :

$$\Delta w = -\eta \frac{\partial J}{\partial w}$$

with η the learning gain which influences the weights convergence speed.

Applying this algorithm to the network weights, we obtain the gradient vectors denoted δ_W and δ_V :

$$\begin{cases} \delta_W = \left(y^d - y\right) * F'\left(W\left(F\left(Ve + V_b\right)\right) + W_b\right) \\ \delta_V = \left(W^T \delta_W\right) * F'\left(Ve + V_b\right) \end{cases} \tag{10}$$

with

$$F'(NET) = \left[f'\left(net_1\right) \quad f'\left(net_2\right) \quad \cdots \quad f'\left(net_{2^m}\right)\right]^T$$

the derivative of $F(NET)$, where $f'\left(net_z\right)$ is the derivative of f with respect to net_z :

$$f'\left(net_z\right) = \frac{\partial f\left(net_z\right)}{\partial net_z} = \frac{e^{-net_z}}{\left(1 + e^{-net_z}\right)^2} \tag{11}$$

$*$ denotes the HADAMAR product.

Thus, the adaptation laws are :

$$\begin{cases} W_{new} = W_{old} + \eta \delta_W \left[F\left(Ve + V_b\right)\right]^T \\ W_{b,new} = W_{b,old} + \eta_b \delta_W \\ V_{new} = V_{old} + \eta \delta_V [e]^T \\ V_{b,new} = V_{b,old} + \eta_b \delta_V \end{cases} \tag{12}$$

The learning gain for the bias weights is η_b with $\eta_b < \eta$, so that their variations are not too large with respect to the weight variations of W and V.

At each sampling time k, the weight matrices W, W_b, V and V_b are updated with the adaptation laws (10) and (12) for all the input - output pairs of set S, even if only one pair $\left(\left\{\Delta x_p\right\}_i, \left(y^d\right)_i\right)$ of S is modified at the kth instant. The interest of using all the training data is to accelerate weight adaptation and to improve the learning rate of convergence. In addition, the advantage is to reinforce the learning of the network and the classification of the 2^m configurations. So the disappearance or the reduction of one class is going to be avoided. With this training strategy, the presented network can be successfully used to control switching systems.

Initialization and Learning Procedure

step 1 : for k = 0.

- If learning has never been performed, then initialize weights randomly. Otherwise load the previous weights.
- Measure $x_p(k)$.
- Set i=1 and apply $Config_i(u)$ during the sampling time T_e.

step 2 : for $k = 1$ to $2^m r - 1$.

- Measure $x_p(k)$ and compute the variation $\left(\Delta x_p\right)_i(k)$.

- Update the average $\overline{\left\{\Delta x_p\right\}_i}$, and set S with the new input-output pair$\left(\overline{\left\{\Delta x_p\right\}_i}, \left(y^d\right)_i\right)$.

- Set $i = k \bmod 2^m + 1$ and apply $Config_i(u)$.

step 3 : for $k \geq r2^m$

- Measure $x_p(k)$ and compute the variation $\left(\Delta x_p\right)_i(k)$.

- Update the average $\overline{\left\{\Delta x_p\right\}_i}$, and set S with the new input-output pair$\left(\overline{\left\{\Delta x_p\right\}_i}, \left(y^d\right)_i\right)$.

- Update the network weights using the updating equations (10) and (12) for the 2^m training pairs of set S.

- Go to the control procedure to determine the control $Config_i(u)$.

Control Algorithm

This Section presents the control law, whose aim is to reduce the error between the desired and the actual state of the system, by finding out the right configuration to apply. As previously established, the neural network is defined to provide the relation between every configuration and the system state variations. Consequently, the learned neural network performs the classification of the 2^m configurations depending on the input vector e.

Definition : Given an n input vector e, it belongs to the class denoted C_i if y_i is the component of maximum value of the output vector y.

$$e \in C_i \quad \text{if} \quad y_i > y_j, \forall i \neq j$$

with $\quad y = F\left(W\left(F\left(Ve + V_b\right)\right) + W_b\right)$

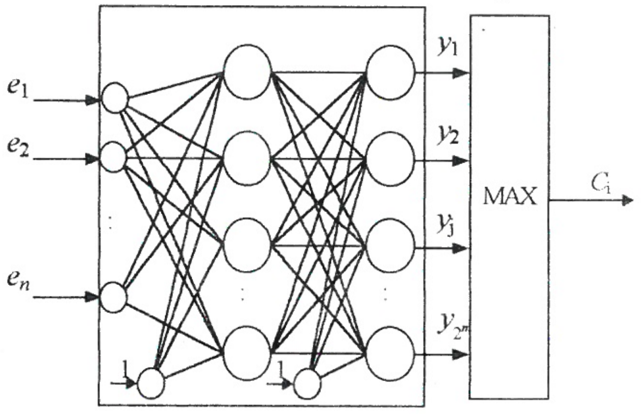

Figure 5. Determination of the *ith* Class

An example of the classification performed by a neural network with two inputs and four classes is given below :

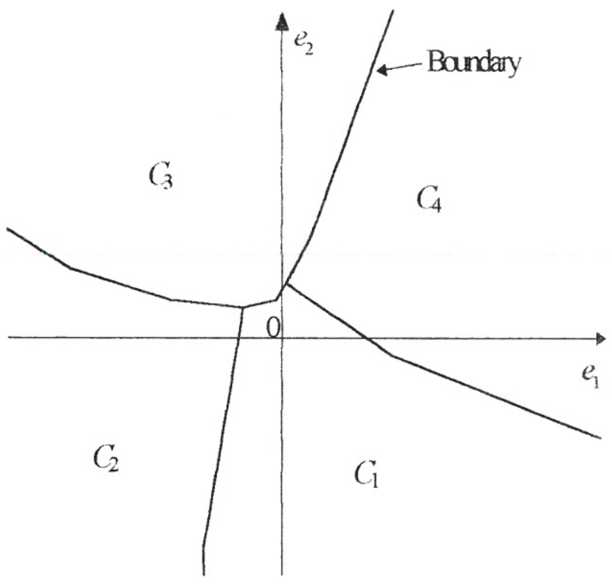

Figure 6. Neural Network Classification

$i = \{1 \cdots 4\}$

This classification is used to choose the configuration which best reduces the tracking error, which is written as follows :

$$\varepsilon = x_d - x_p \tag{13}$$

where x_d is the desired state and x_p the process state.

By then applying this vector ε to the network input, we obtain an output vector y whose component of maximum value gives the class of ε. Knowing this class, we can deduce the matching configuration and the associated control vector.

Control Procedure

step 1 : Compute the position error ε

step 2 : Compute the network output y with ε as input.

step 3 : Determine the maximal element y_i of y to deduce the right class i to choose.

step 4 : Select the control vector $Config_i(u)$

step 5 : Apply the control vector to the switching system during the sampling time T_e.

4. Application to A Synchronous Motor

Presentation

The application has been performed in order to evaluate the performance of the proposed Boolean control algorithm. The system used to obtain the simulation results is composed of a synchronous machine and a power converter (Figure 7). The power converter is constituted of six switching transistors (S_1, S_2, S_3, S_4, S_5, S_6) and supplied by a continuous voltage E_o. The synchronous motor design comprises 3 wires to the stator and 3 wires to the rotor.

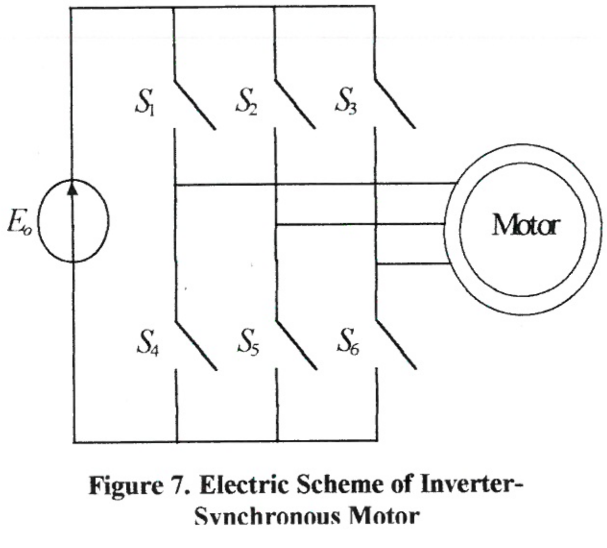

Figure 7. Electric Scheme of Inverter-Synchronous Motor

In this approach of the problem of motor control, a mathematical model of the system is required to simulate its behavior. By taking the statoric currents, the angle and the velocity as state variables the mathematical model of the system [13] is given by :

$$\begin{cases} \dfrac{di_\alpha}{dt} = -\dfrac{R}{L} i_\alpha + \dfrac{\phi_f}{L}\omega \sin(\theta) + \dfrac{1}{L} u_\alpha \\ \dfrac{di_\beta}{dt} = -\dfrac{R}{L} i_\beta - \dfrac{\phi_f}{L}\omega \cos(\theta) + \dfrac{1}{L} u_\beta \\ \dfrac{d\theta}{dt} = \omega \\ \dfrac{d\omega}{dt} = -\dfrac{f}{J}\omega + \dfrac{\phi_f}{J}\left(i_\beta \cos(\theta) - i_\alpha \sin(\theta)\right) - \dfrac{C_r}{J} \end{cases} \tag{14}$$

i_α, i_β are the equivalent statoric currents after the reduction of the model. θ represents the rotor angle relative to a statoric fixed reference frame. ω is the velocity of the synchronous machine. ϕ_f represents rotoric flux amplitude.

u_α and u_β are the voltages applied to the stator. These voltages are determined by Boolean values such as :

$$\begin{cases} u_\alpha = \dfrac{E_o}{\sqrt{6}}(2u_1 - u_2 - u_3) \\ u_\beta = \dfrac{E_o}{\sqrt{2}}(u_2 + u_3) \end{cases}$$

The different items u_i for $i = 1,2\cdots6$ are the Boolean associated with the six transistors of switches (S_1, S_2, ..., S_6). $u_i = 0$ if the i^{th} transistor S_i is in the OFF state and $u_i = 1$ if it is in the ON state. For electrical reasons, they are linked by the relation :

$$u_i + u_{i+3} = 1 \text{ for } i = 1,2,3$$

In all that follows $u = \begin{bmatrix} u_1 & u_2 & u_3 \end{bmatrix}^T$ is considered as the input vector of the system.

The other parameters are defined as :

R : Statoric resistance

L : Statoric inductance

E_o : Voltage applied to the power converter

f, J : Friction and inertia of the rotor

C_r : Opposing torque

Our aim is to control the synchronous machine in order to obtain a desired velocity ω_d [14]. The tuning of the velocity ω of the synchronous machine is performed by using a non-linear velocity equation.

$$\frac{d\omega}{dt} = -\frac{f}{J}\omega + \frac{\phi_f}{J}\left(i_\beta \cos(\theta) - i_\alpha \sin(\theta)\right) - \frac{C_r}{J} \tag{15}$$

To control the system to the desired angular speed ω_d, we can show that it is sufficient to control the statoric currents, and to provide them as two waveform sinusoidal reference currents defined as in Eq 16 .

$$\begin{cases} i_\alpha ref = -i_o \sin(\theta) \\ i_\beta ref = i_o \cos(\theta) \end{cases} \tag{16}$$

By substituting these reference currents into Eq (15), we find :

$$\frac{d\omega}{dt} = -\frac{f}{J}\omega + \frac{\phi_f}{J} i_o - \frac{C_r}{J} \tag{17}$$

It can clearly be deduced from this equation that the velocity regulation depends on the amplitude i_o of the reference currents. For example, for a constant desired speed ($\omega = \omega_d$ and $\frac{d\omega}{dt} = 0$), the amplitude i_o is :

$$i_o = \frac{f}{\phi_f}\omega_d + \frac{C_r}{\phi_f} \tag{18}$$

The scheme proposed for speed control is shown in Figure 8. The motor speed regulation is simply achieved with a feedforward action combined with a well-known proportional-integral (PI) controller. The interest of the feedforward is to improve the dynamic control performance. This compensator will then enable that satisfactory performances are

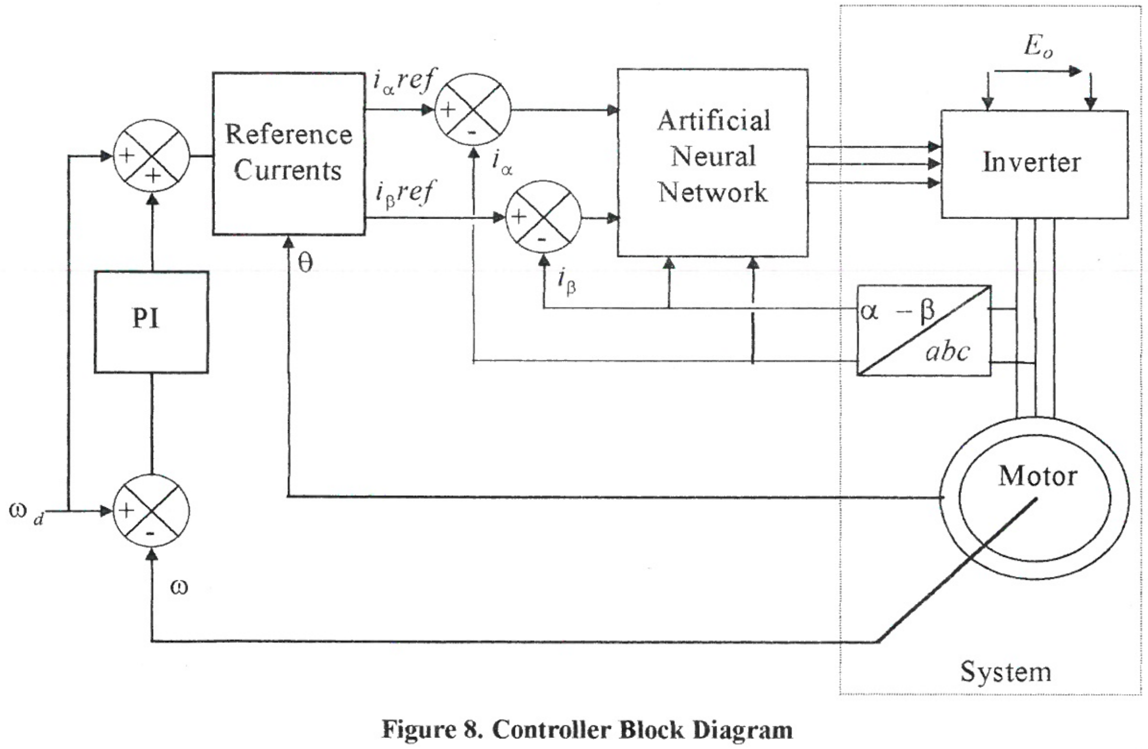

Figure 8. Controller Block Diagram

reached for this application. The gains of the PI are chosen such that the response is fast and with a low overshooting for the nominal conditions. With this controller to reduce the error between the reference signal ω_d and the process output ω, the control law of the current i_o is expressed by :

$$i_o = \frac{f}{\phi_f}\left(\omega_d + K(\omega_d - \omega) + \frac{1}{T_i}\int_0^t (\omega_d - \omega)dt\right) + \frac{C_r}{\phi_f} \tag{19}$$

For the application concerned, the PI controller gives satisfactory results with the gains chosen as follows :

$$K = 5 \text{ and } T_i = 2.$$

The reference currents $i_\alpha ref$ and $i_\beta ref$ of statoric wires are generated by the relationships (16) and (19), previously described.

The phase currents i_a, i_b and i_c are measured and reduced in two components of current i_α and i_β of an equivalent two-phase machine. These components are compared to the reference currents $i_\alpha ref$ and $i_\beta ref$. The outputs of these comparators are the input controls ε of the neural network, and the currents i_α and i_β are the inputs for the learning of this network. The control process is now constituted by two regulation sub-systems, one for the speed control, and the other for tuning the currents.

The main objective of this work is that the system tracks a trajectory in the state space. The desired trajectory is imposed here by the current references $i_\alpha ref$ and $i_\beta ref$. This objective is achieved by using the artificial neural network, previously described.

Simulation Results

The MATLAB-SIMULINK simulation software has been used to study the response of the electrical system. The complete drive was simulated in C, modeling the real time dynamics with an $100\mu s$ time step. The equations of the complete drive have been resolved by the fifth Runge-Kutta order method for the numerical integration. The three-phase inverter has been simulated considering ideal switches and the synchronous machine has been simulated by the state equation (14) with the following parameters :

$$\begin{cases} R : 2\,\Omega \\ L : 200\text{ mH} \\ Eo : 380\text{ V} \\ f : 0.08\text{ N.s} \\ J : 0.02\text{ kg.m2} \\ Cr : 0.8\text{ N.m} \\ \phi_f : 1\text{ Wb} \end{cases}$$

The structure of the neural network used is a 2-8-8 structure (two inputs, eight neurons in the hidden layer, and eight neurons in the output layer). The sampling time of the neural network is 0.2 ms, which corresponds to a 5 khz switching frequency. This sampling period is chosen according to the dynamics of the system and the frequency limitation of the components.

Due to the inverter structure (Figure 7), the motor behavior for configurations 1 and 8 is the same, which means that data of classes 1 and 8 overlap. In general, classes that overlap or surround each other cannot be separated. Consequently classes 1 and 8 have to be considered as being identical. To overcome this problem, target vectors $(y^d)_1$ and $(y^d)_8$ are chosen as follows:

$$(y^d)_1 = (y^d)_8 = \begin{bmatrix} y_1^d \\ 0 \\ \vdots \\ 0 \\ y_8^d \end{bmatrix} \text{ with } y_1^d = y_8^d = 1 \tag{20}$$

This new definition of target vectors does not influence the neural control algorithm since configurations 1 and 8 are equivalent with respect to the motor drive.

The neural network is trained by the previously presented learning algorithm, and with the above target vectors. The learning rates η and η_b are respectively 0.1 and 0.01, which ensures both a fast and stable weight adaptation.

We have also to take into account the measure noise effect which cannot be neglected in power electronics. Indeed, measures provided by typical sensors are inherently affected by noise, which is generally assumed to be white. Their magnitudes for the current sensor, the position sensor and the speed sensor are respectively supposed to be equal to 0.1 A , 0.005 rad and 0.05 rad/s. To reduce their effect, we use the filtering method defined in the Data Processing Section with the coefficient r=10.

To illustrate the learning procedure, the time required for the network to perform a correct classification of the 2^m configurations is discussed. Under the previously defined simulation conditions, the weights evolve from random initial weights to satisfactory weights in less than 0.4 s. So, in this example, about 2,000 complete iterations are necessary, when performing the learning for the first time. This stage occurs only once since the following network initializations are done with saved weights. Therefore, as shown in Figure 9, the learning procedure based on the backpropagation algorithm, and the control derived from the network yield good results.

After this preliminary learning, the neural network and the overall controller are ready to control currents and the motor speed. Figures 10 and 11 show the actual and the desired currents obtained for a speed reference ramp increasing from 0 up to 40 rad/s in 2 seconds.

In order to provide a quantitative evaluation of the performance and accuracy of the proposed scheme, we calculate the criterion defined as the mean of the absolute magnitude of errors between actual and desired values. Despite the noise measurements and the Boolean switches, currents will converge towards the reference currents precisely. The mean absolute errors for i_α and i_β are respectively about 0.12 A and 0.13 A. Such an accurate regulation of these currents is absolutely necessary to obtain a good speed regulation. We further examine the ability of the proposed controller to track a specific desired speed reference. The results are shown in Figure 12.

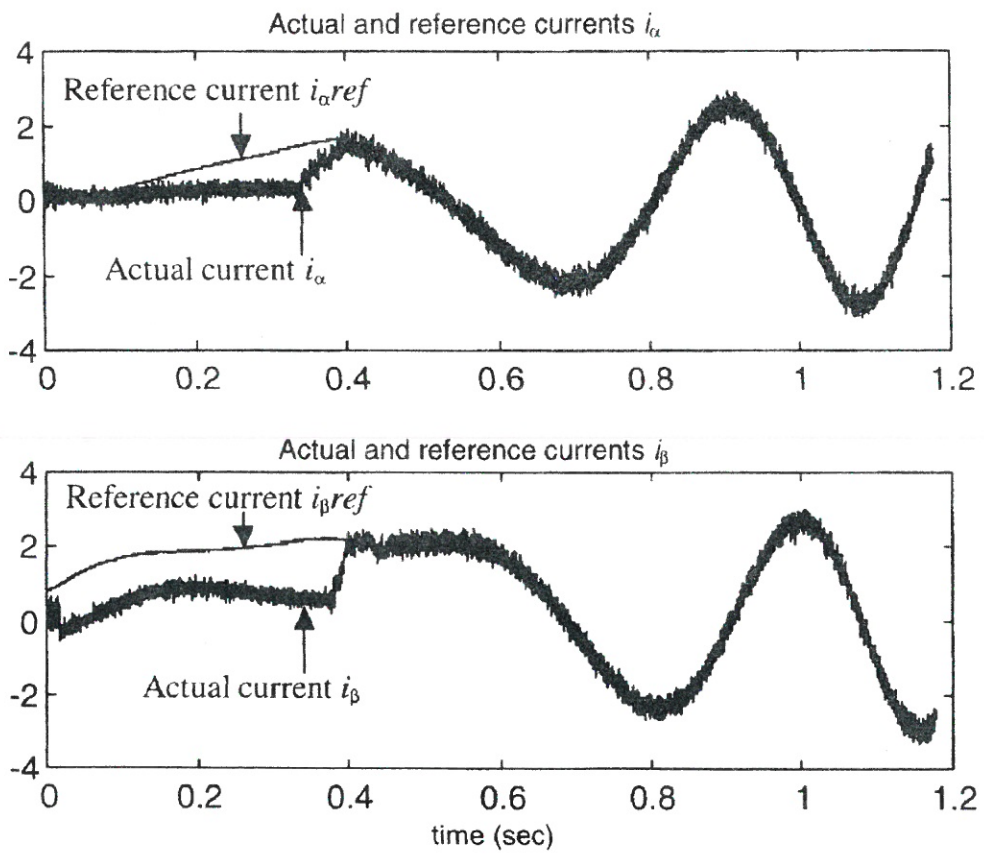

Figure 9. Actual and Reference Currents for the First Learning

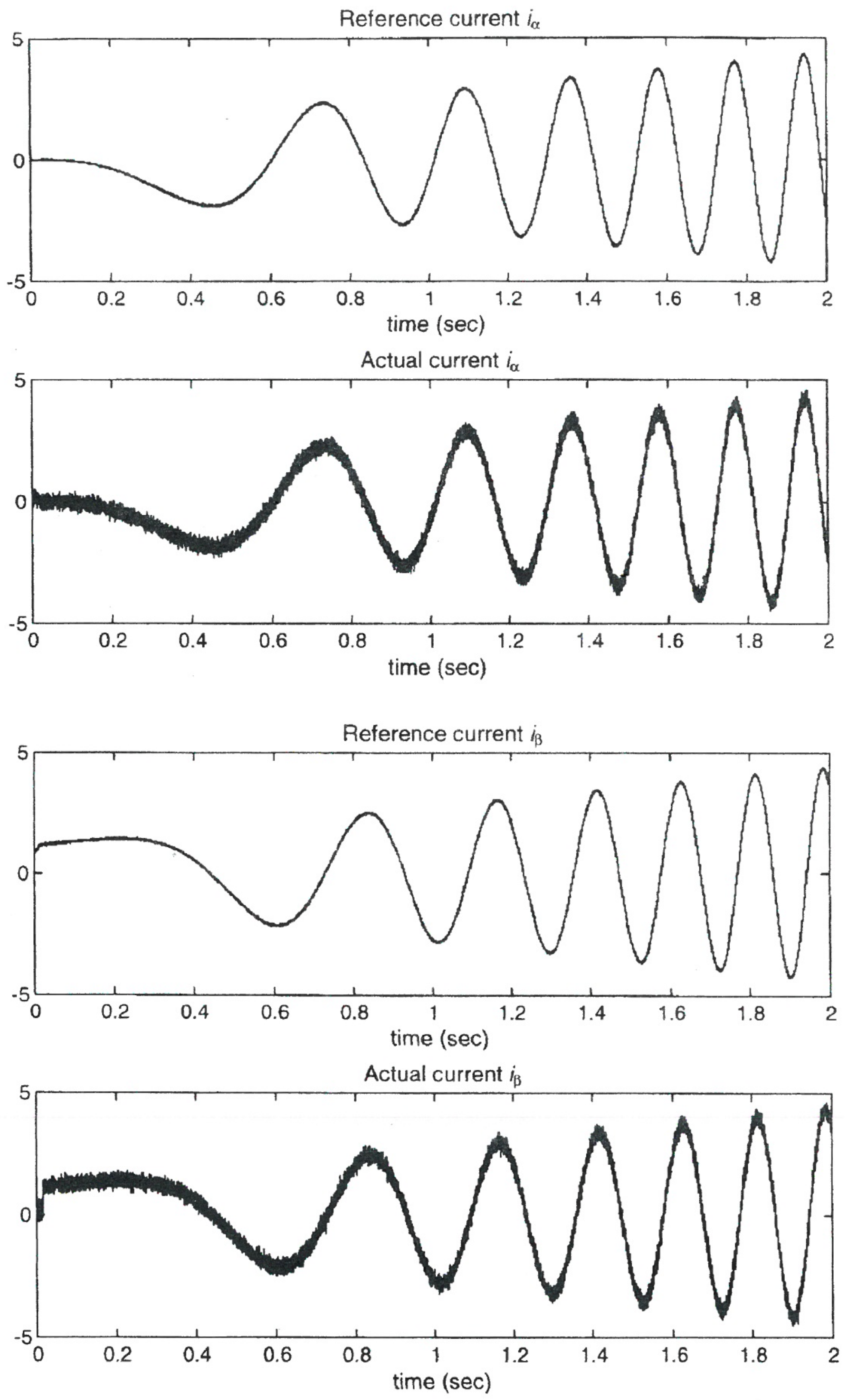

Figure 10. Actual and Reference Currents for A Ramp Input

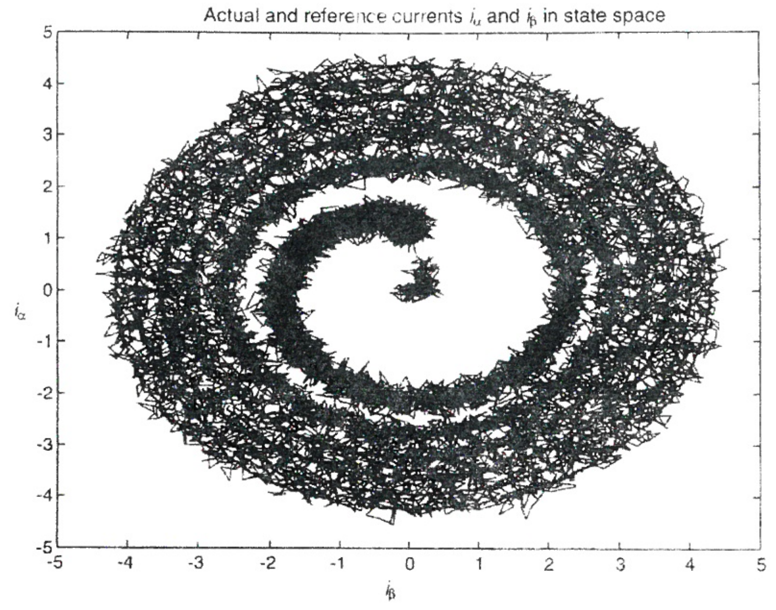

Figure 11. Actual and Reference Currents in State Space for A Ramp Input

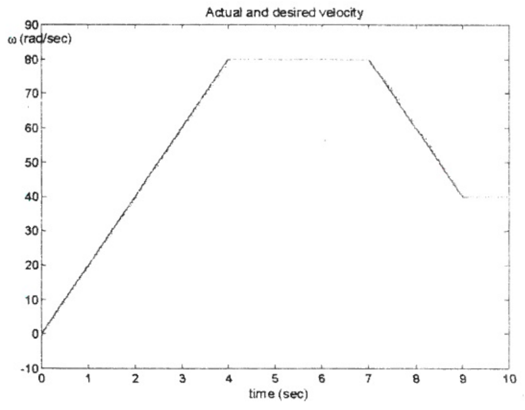

Figure 12. Actual and Reference Speed

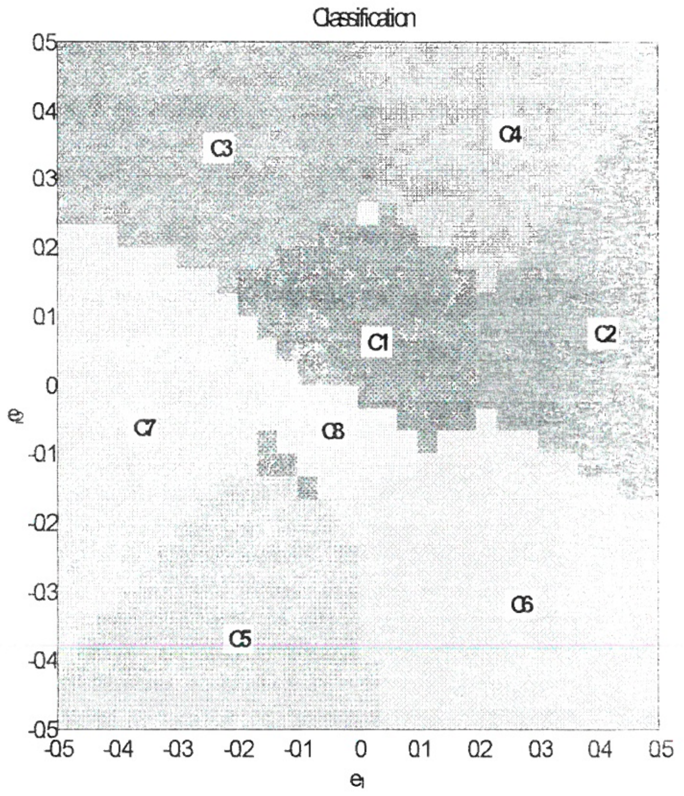

Figure 13. Neural Classification

The desired speed ω_d is followed with a maximum absolute error lower than 1.1 rad/s. The mean absolute error over the simulation interval is 0.51 rad/s. However, between the 4th and the 7th seconds (ω_d = 80 rad/s), the mean absolute error is only about 0.05 rad/s. It is thus possible to verify on the speed tracking being efficient and accurate.

The network analysis is achieved by mapping classes with respect to the input variables e. In the center of Figure 13, we notice that classes 1 and 8 overlap as explained before. The other 6 classes are well separated and, as shown in Figure 14, the class boundaries are clearly sharp. Indeed the network outputs y_i vary swiftly between the low level 0 and the high level 1 of activation.

These results exhibit the ability of the network to perform a good classification of the 2^m configurations. The Boolean control law based on this network therefore provides high and consistent performance.

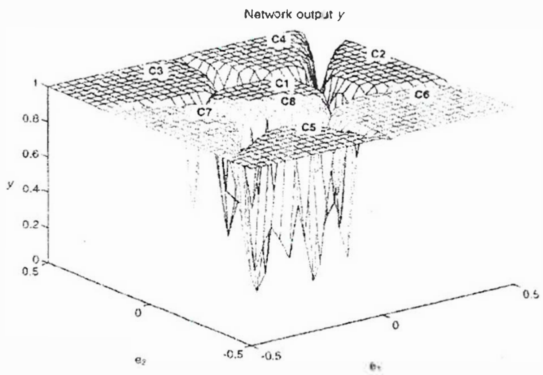

Figure 14. Maximum Network Outputs

5. Conclusion

In this paper a method of control using Artificial Neural Network is proposed for systems with Boolean inputs. An appropriate neural network structure with an adequate initialization method and efficient data processing have been developed to deal with the problem of controlling nonlinear switching systems.

The initialization step and the data filtering give, despite the measurement noise, a set of satisfactory training data . These procedures are necessary to improve the network learning. This latter is based on the backpropagation algorithm, which is a relatively low cost computing method, and so, it is easy to implement in real time. This learning algorithm is applied on-line, at each sampling time, to the complete training data set. The obtained advantage is an increase in the weight convergence rate and the reinforcement of the classification. Finally, this classification is used to deduce the Boolean control law.

The effectiveness of this approach is confirmed by simulation results obtained with a synchronous motor as application system. The dominance of the neural network is to control the system without exact knowledge of it. Online adaptation considerably improves the robustness of the system with respect to parametric changes. In conclusion, the proposed artificial neural network shows high performance and good control accuracy for switching systems.

REFERENCES

1. RAHMAN, M. F. and ZHONG, L., **A Current-forced Reversible Rectifier Fed Single-Phase Variable Speed Induction Motor Drive**, Proceedings of PESC, Vol. 1, 1996, pp. 114-119.
2. ISMAIL, E.H. and ERICKSON, R., **Single-Switch 3ϕ PWM Low Harmonic Rectifiers**, IEEE POWERS ELECTRONICS, March 1996, pp. 338-346.
3. NONAKA, S. and NEBA, Y., **Analysis of A PWM GTO Current Source Inverter Fed Induction Motor Drive System**, IEEE TRANSACTIONS ON INDUSTRY APPLICATIONS, Vol. 23, No 2, 1987.
4. SIRA-RAMIREZ, H., **Sliding Regimes in General Non Linear Systems : A Relative Degree Approach**, INT. J. CONTROL, Vol. 50, No. 4, 1989, pp.1487-1506.
5. SLOTINE, J. E., **Sliding Controller for Non-linear Systems**, INT. J. CONTROL, Vol. 40, No. 2, 1984, pp. 421-434.
6. SABANOVIC, A. and IZOSIMOV, D. B., **Application of Sliding Modes to Induction Motor Control**, IEEE TRANSACTIONS ON INDUSTRY APPLICATIONS, Vol. 17, No. 1, 1981, pp. 41-49.
7. JOHNSON, G. E., **Mimic Nets**, IEEE TRANSACTIONS ON NEURAL NETWORKS, Vol. 4, No. 5, September 1993, pp. 803-815.
8. PATTERSON, D. W., **Artificial Neural Networks**, PRENTICE HALL, Singapore, 1996.
9. NARENDRA, K. S. and PARTHASARATHY, K., **Identification and Control of Dynamical Systems Using Neural Networks**, IEEE TRANSACTIONS ON NEURAL NETWORKS, Vol. 1, No. 1, 1990.
10. ABADIE, V. and DAUPHIN-TANGUY, G., **Opened Loop control of switching linear system**, JOURNAL OF THE FRANKLIN INSTITUTE, Vol. 330, No. 5, 1993,pp. 799-813.
11. HOLDERBAUM, W., Dauphin-Tanguy, G . and Borne, P., **Boolean Control for Linear System**, Proceedings of International Symposium on Intelligent Automation and Control, ISIAC WAC'98 Anchorage, USA, 1998, pp. 212.1-212.6 .
12. HOLDERBAUM, W., DAUPHIN-TANGUY, G .and BORNE, P., **Tracking Control Problem for Switching Linear System**, Proceedings of CESA'98 IMACS-IEEE/SMC Conference, Hammamet Tunisia, Vol. 1, 1998, pp. 935-940.
13. DUCREUX, J. P., CASTELAIN, A., DAUPHIN-TANGUY, G. and ROMBAUT, C., **Power Electronics and Electrical Machines Modelling Using Bond-Graph** , in G. Dauphin-Tanguy and P. Breedveld (Eds.) IMACS TRANSACTIONS ON BOND-GRAPH FOR ENGINEERS, Elsevier, NewYork, 1992.
14. LEONHARD, W., **Control of Electrical Drives**, SPRINGER, Berlin , 1985.