In: Computational Intelligence and Applications, P. S. Szczepaniak (ed.), (Physica-Verlag, c/o Springer-Verlag, 1999).

MODELING OF ELECTROMECHANICAL SYSTEMS USING NEURAL NETWORKS

L. Constant *, P. Lagarrigues *, B. Dagues *, I. Rivals **, L. Personnaz **
Laboratoire d'Electrotechnique et d'Electronique Industrielle de Toulouse (LEEI) (*)
Laboratoire d'Electronique de l'Ecole Supérieure de Physique et de Chimie Industrielles de Paris (ESPCI) (**)

Abstract

We present a new model of the induction machine based on neural network theory. After a brief presentation of the neural modeling methods used in this work, we introduce the neural model architecture, which is based on the Park model. We then describe the training procedure of the neural model, and give evaluations of its performance, for example on startups with a speed vector control.

1. Introduction

This paper deals with a new approach for the modeling of electromechanical conversion devices based on neural networks. The design of an emulator of the process {static converter - electrical machine - sensors} involves the simulation of complex electrotechnic systems in real time [Ben Saoud et al., 1996]. Classic implementations of induction machine (IM) models based on the Park transform are computationnally intensive, which prevents real time simulation. This paper describes the design of a dynamic neural model (NM) of the IM electromagnetic part. As a matter of fact, the ability to approximate nonlinear mappings parsimoniously and the possibility of parallel computing make neural networks efficient in terms of accuracy and computation time. The NM we use is a feedback multi-layer network whose global structure is based on the Park model; this NM should thus be considered as a complement or an emulator of the latter. The NM is implemented with the C function library Simul_NN developed at the ESPCI, and the Park model with the software PostMac created by the LEEI.

2. Neural networks for dynamic process modeling

The goal of modeling a process is to build a mathematical model which emulates its dynamic behavior, given some prior knowledge about the process and input-output measurements [Rivals & Personnaz, 1996]. It is usually distinguished between knowledge-based and black-box modeling, according to the amount of prior knowledge about the internal behavior of the process that is available to build a model:

- A modeling procedure is termed knowledge-based if the model is built from physical insight, where the only unknowns are parameters which are estimated from measurements. Knowledge-based models are usually in the state-space form, their state variables and the relationships between them having a physical meaning, like the Park model.
- If insufficient physical insight is available, but only observed data, one must resort to black-box modeling. A discrete-time black-box model is a recursive filter whose outputs are parameterized functions of its past external inputs and state variables. In order to be able to describe any complex dynamic process, the family of parameterized functions should be as flexible as possible, like neural networks [Hornik et al., 1989].

The fact that this paper deals with neural networks might suggest that we are dealing with black-box modeling; actually our approach is partly knowledge-based, partly black-box. As a matter

of fact, the process we model is not, at least not in a first step, the IM itself, but the PostMac simulator of the IM, whose computations are based on the Park transform. We thus have entire knowledge about the process, and are free to use it to build a model. Our motivations for the use of this knowledge are the following:

- A reliable model of a complex process like the IM cannot be entirely obtained from input-output data. Even if a good input-output model could be designed, it is of interest for control purposes that state variables like the magnetic fluxes be available. Our aim is thus to build a model with the same inputs, state variables and outputs as the PostMac simulator. This leads to a modular modeling approach.
- We want to make use of the parsimony and of the possibility of parallel computing of neural networks. Therefore, we will not try to copy the structure of each module, but to emulate it with a black-box neural network.
- The modular approach opens the way to the modeling of more complex behaviors of the IM (with magnetic saturation for example), this time from measurements on a real IM.

This modular approach leads to the following procedure. Each module can be considered as a separate process in the input-output form:

$$Y_p(k) = f\left(Y_p(k-1), \ldots, Y_p(k-n), U(k-1), \ldots, U(k-m)\right) \tag{1}$$

where $Y_p(k)$ is the output vector and $U(k)$ is the input vector at time k. The process is noiseless since it is simulated. The goal of the modeling procedure is to build a NM which predicts the process output $Y_p(k)$ as accurately as possible, given the past inputs and outputs. Theoretically, this NM can be either a feedforward one:

$$Y(k) = \varphi\left(Y_p(k-1), \ldots, Y_p(k-n), U(k-1), \ldots, U(k-m); \theta\right) \tag{2}$$

or a feedback one:

$$Y(k) = \varphi\left(Y(k-1), \ldots, Y(k-n), U(k-1), \ldots, U(k-m); \theta\right) \tag{3}$$

where φ is the nonlinear function implemented by a neural network with weights θ. In order to adjust the weights of a model with a given architecture (number of NMs and connectivity) so that φ becomes as close as possible to f, the most common method consists in minimizing iteratively a quadratic cost function of the prediction errors, cost function which is defined over a training sequence of input-output pairs $\{U(k), Y_p(k)\}_{k=1 \text{ to } N}$. At iteration i, its value is:

$$J(\theta^i) = \sum_{k=1}^{N} \left(E^i(k)\right)^T W E^i(k) = \sum_{k=1}^{N} \left(Y_p(k) - Y^i(k)\right)^T W \left(Y_p(k) - Y^i(k)\right) \tag{4}$$

where W is a weighting definite positive matrix, and $Y^i(k)$ is the output of the model at time k and iteration i, i.e. with weights θ^i. The cost function $J(\theta^i)$ is minimized by a gradient descent followed by a quasi-newtonian method. The gradient is computed either in a directed fashion [Nerrand et al., 1993], i.e. using the Teacher Forcing algorithm [Jordan, 1985], or in a semi-directed fashion, i.e. using the Backpropagation-Through-Time algorithm [Rumelhart et al., 1986]. The universal approximation property of neural networks [Hornik et al., 1989] guaranties that φ can be arbitrarily close to f, provided the network has a sufficient architecture. To select a network architecture, the performance of the network models are estimated on a validation sequence, in a feedback fashion since the performance of interest is not only a one-step-ahead prediction, but the simulation over a large time-horizon. If the training and validation sequences are chosen according to the future use of the model (type of inputs, sample period, etc.) and if the training algorithm is efficient, then the training and selection procedures will lead to a model whose output is arbitrarily close to the process output in the domain covered by the training and validation sequences.

3. From the Park model to the neural model structure

A stator fixed reference frame is used for the Park transform, which leads to a system of four differential equations governing the behavior of the fluxes ϕ_{sd}, ϕ_{sq}, ϕ_{rd} and ϕ_{rq} [Caron & Hautier, 1995] [Notelet & Seguier, 1994]. They depend on the Park stator voltages V_{sd} and V_{sq}, and on the rotor speed Ω. This system is modeled by the feedback part of the NM shown in Figure 1. Since the stator fluxes are linearly dependent on the inputs, each of them is computed by one linear neuron. The Park model shows that the rotor fluxes are nonlinear functions of the inputs: the computation of each rotor flux thus necessitates a nonlinear network (with nine sigmoidal hidden neurons and one linear output neuron). The structure of the static part of the model is settled in the same way. Each Park current (I_{sd}, I_{sq}) is computed by a linear neuron whose inputs are the corresponding Park fluxes. The module which computes the electromagnetic torque is a nonlinear network (with ten sigmoidal hidden neurons and one linear output neuron). The optimal number of hidden neurons in each module was established according to the results of trials with an increasing number of neurons.

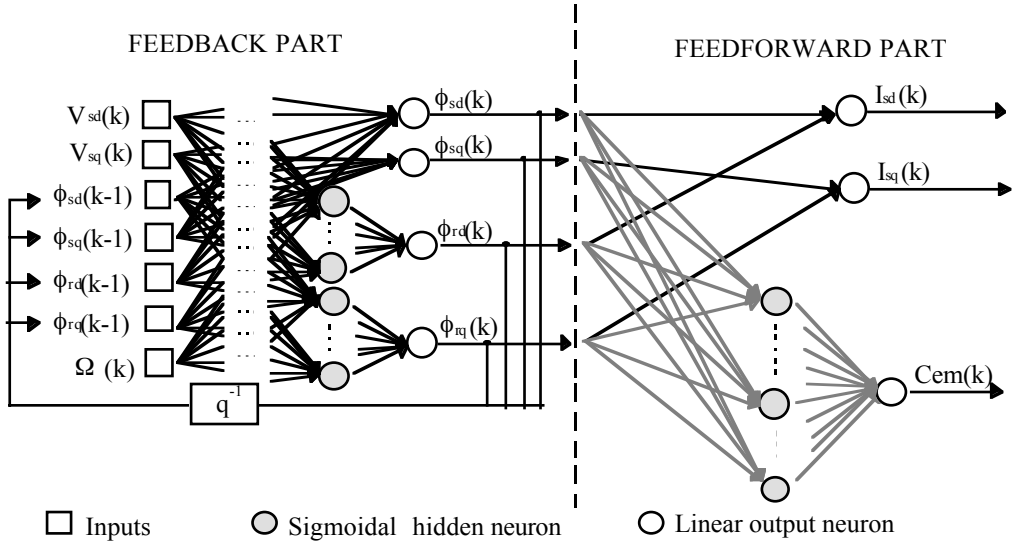
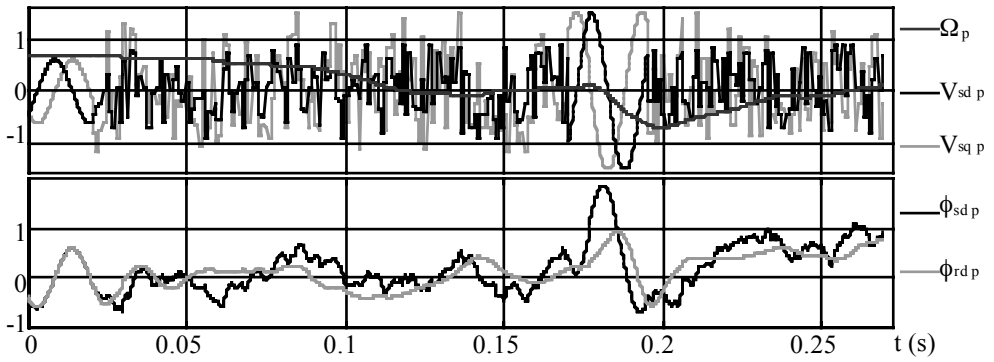


***Figure 1**: Neural model structure*

4. The training of the neural model

4.1. Choice of the training sequence

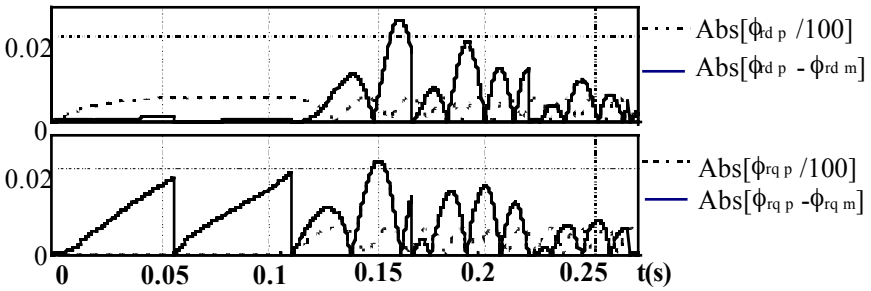
In order to estimate the parameters of the NM, we have to build up a training sequence which describes the behavior of the PostMac simulator of the IM, i.e. the "process" variables must explore its whole operating range. The duration of the training sequence is 0.27 second with a sampling period of 10 μs (27000 successive samples for each variable). To facilitate the training, all variables are set to the same scale.



Graph 1: *Inputs and fluxes $\phi_{sd\,p}$ and $\phi_{rd\,p}$ of the training sequence*

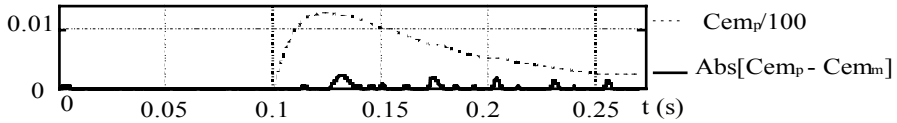
4.2. Training each separate neural module

We first consider the feedback part of the NM (Figure 1). Each flux is computed by an independent module of the complete NM. We start by training these modules in a directed fashion (i.e. the output of each module is computed from the process flux), and improve their accuracy by training them in a semi-directed fashion (i.e. the module output is computed from its previously computed value). Finally we assemble the four modules to train the complete flux model in a semi-directed fashion. Graph 2 shows the rotor flux error on a validation sequence. The square root of the mean-square-errors (RMSE) of the four fluxes is $8.1\ 10^{-3}$.



Graph 2: *Rotor flux error on a validation sequence (IM start)*

We use the same training sequence to train the feedforward part of the complete NM, which is composed of three independent modules, one for each stator current and one for the electromagnetic torque. All of them are feedforward models. After training, the value of the two currents RMSE is $4.15\ 10^{-7}$ on the validation sequence, and the value of the electromagnetic torque RMSE is $8.72\ 10^{-4}$.



Graph 3*: Electromagnetic torque error on a validation sequence (IM start)*

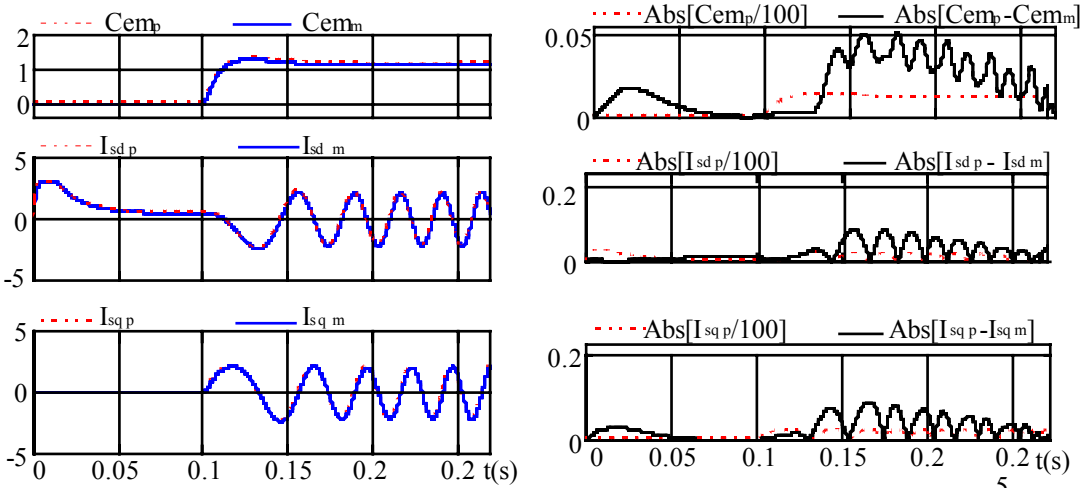
4.3. Training the complete neural model

We can now assemble all modules to obtain the complete NM described in Figure 1. A validation sequence is build as an off-load start with a speed vector control: a rotor flux setting at zero speed followed by a speed step response. On this validation sequence, the value of the global RMSE (i.e. of the four fluxes, the electromagnetic torque and the two currents) is 7.43 10^{-2}. To improve the accuracy of the model, we continue the training on a new training sequence obtained by adding the validation sequence to the previous training sequence. After

training, we test the NM on three new validation sequences: three machine startups with a mechanical load, and three different reference speeds (see Table 1).

Reference speed	$0.5*\Omega_n$	Ω_n	$1.5*\Omega_n$
Global RMSE	$5.95\ 10^{-2}$	$4.75\ 10^{-2}$	$2.97\ 10^{-2}$

***Table 1**: Global RMSE on validation sequences (Ω_n: nominal speed)*



***Graph 4**: Validation test at the nominal speed*

5. Conclusion

Our results already demonstrate the feasibility of the accurate modeling of an induction machine with feedback neural networks. Nevertheless, these results can be improved by the use of more extensive training sequences, especially at low speed. An important advantage of neural modeling is that the knowledge of the electrical parameters of the machine is not necessary, but only input-output data. Our model can thus be adapted to the modeling of a machine with magnetic non-linearities, provided that suitable training sequences are available (either simulated or measured).

The aim of our next study is to associate our neural induction machine model to a static converter model, in order to implement this global model on a real experimental device, and to compare its rapidity to that of a classic Park model.

Notations

V_{sd}, V_{sq}	*Park stator voltages*
I_{sd}, I_{sq}	*Park stator currents*
ϕ_{sd}, ϕ_{sq}	*Park stator fluxes*
ϕ_{rd}, ϕ_{rq}	*Park rotor fluxes*
Cem	*electromagnetic torque*
Ω	*rotor speed*
p	*process variable index*
m	*neural model variable index*

References

BEN SAOUD S., DAGUES B., SCHNEIDER H., METZ M., HAPIOT JC., 1996, *Real time emulator of static converters / electrical machines- Application to the test of control unit,* ISIE' 96, 17-20 June, 1996, WARSAW, Poland

CARON J.P., HAUTIER J.P., 1995, *Modelisation et commande de la machine asynchrone*, Méthodes et pratiques de l'ingénieur, Technip.

HORNIK K., STINCHCOMBE M., WHITE H. (1989), *Multilayer feedforward networks are universal approximators*, Neural Networks **2**, pp. 359-366.

JORDAN M. I. (1985), *The training of representations for sequential performance*, Doctoral Dissertation, University of California, San Diego.

NERRAND O., ROUSSEL-RAGOT P., PERSONNAZ L., DREYFUS G. (1993), *Neural networks and nonlinear adaptive filtering: unifying concepts and new algorithms*, Neural Computation **5**, pp. 165-199.

NOTELET F., SEGUIER G., (1994), *Electrotechnique industrielle,* Lavoisier TEC&DOC.

RIVALS I., PERSONNAZ L., 1996, *Black-box modeling with state-space neural networks*, in Neural Adaptive Control Technology, R. Zbikowski and K. J. Hunt eds.,World Scientific, pp. 237-264.

RUMELHART D. E., HINTON G. E., WILLIAMS R. J. (1986), *Training internal representations by error back-propagation*, in Parallel Distributed Processing: explorations in the microstructure of cognition. Vol.1 : Foundations, D. E. Rumelhart, J. L. McLelland and the PDP Research Group eds, MIT Press, Cambridge MA, pp. 318-362.