International Journal of Sciences and Techniques of Automatic control & computer engineering
IJ-STA, Volume 5, N° 1, June 2011, pp. 1500–1515.

A Multimodel Emulator For Non Linear System Controls

Nesrine BAHRI; Anis MESSAOUD & Ridha BEN ABDENNOUR

Research Unit: Numerical Control of Industrial Processes
National School of Engineers of Gabes
Route de Medenine, 6029 Gabès, Tunisie
University of Gabes
bahri.nesrine@gmail.com messaoud_anis@yahoo.fr
Ridha.benabdennour@enig.rnu.tn

Abstract. This work describes a multimodel emulation of a nonlinear system dynamics. Indeed, in the presence of a strong non-linearity, the classical emulation strategy, based on a neural network, does not always lead to a good closed loop performance. An uncoupled multimodel emulator is proposed for the emulation of nonlinear systems. The effectiveness of the proposed emulator is shown through a numerical simulation. The obtained results are very satisfactory and show a very good precision relatively to the case where the classical neural emulator is adopted.

Keywords: Multimodel Emulator, Uncoupled multimodel, Nonlinear System Control.

1 Introduction

Technological development is steadily increasing the complexity of systems. Indeed, nonlinear models are widely used in engineering science applications to describe the dynamic behaviour of real-world processes [3]. Due to their mathematical complexity, the last models are not easily exploitable. To apply, in this case, a control law that ensure consistent good performance, it is necessary to further adjust the controller parameters on-line. Therefore various methods have been proposed for adjusting the parameter values of some control systems [8]. However, this methods are usually based on algorithms that require a precise knowledge of the process dynamics and which need to evaluate the output variation against the input one. But, this last evaluation is not often easy, especially when the system has complex nonlinear dynamics. The design of an emulator for the system dynamics is, therefore, very necessary [5, 17].
In this regard, classical control approaches proposed neural networks as a plant emulator for solving the problem related to the determination of the process dynamics [1, 2, 5, 17]. This approach previously proposed and developed by the authors, has been implemented for decadal climate model simulations. It

This paper was recommended for publication in revised form by the editor Staff.
Edition: CPU of Tunis, Tunisia, ISSN: 1737-7749

has proved to be efficient and successful for different applications. Moreover, for systems with simple non-linearity, acceptable performances may be maintained with neural emulator. However, under certain conditions, the neural emulator can present a relative inability to ensure good performance in closed loop. Indeed, in the presence of disturbances and when the plant presents a strong nonlinearity, the use of this emulator, is not suitable and decrease considerably the performances. To overcome this problem, we propose in this paper an uncoupled multimodel emulator. This emulator is obtained by using a model's library which result from an off-line identification procedure of nonlinear systems using an uncoupled state multimodel approach [11, 13, 14].

The multimodel approach is an interesting alternative and a powerful tool for modeling and controlling complex processes.
This approach is useful for industrial processes which are, often, nonlinear and/or non stationary. The basic idea of this approach is the decomposition of the full operation range of the process into a number of operating regimes. In each operating regime, a simple local model is applied. There are various ways of connecting these local models in order to yield the global model. We can distinguish two multimodel structures according the use of coupled or uncoupled states.
The coupled state structure is the more classically used in multimodel analysis and synthesis [6, 7, 9, 10, 11, 16, 18, 19]. On the other hand, the identification procedure based on the uncoupled state multimodel has been not much used [4, 13, 14].

In this paper, we propose a multimodel emulator based on the uncoupled multimodel to emulate and control non linear processes. Firstly, we present the classical neural emulator. An example of simulation is given thereafter to show the limits of this emulator used in PID Self-Tuning control method. In a second part, we propose a solution for these problems through the synthesis of a multimodel emulator. In this part we give a description of the parametric estimation procedure for uncoupled state multimodel. Then we develop the uncoupled multimodel emulator. A simulation example is proposed, illustrating that the proposed emulator is more precisely and presents good closed loop performances by comparison with the neural emulator.

2 Classical Neural Emulator

Many researchers have considered the emulation problem, using neural network [1, 2, 5, 15, 17]. In all of these studies the researchers have considered that multilayer back-propagation neural networks can be trained to emulate any complicated dynamics.
To obtain the precise dynamics of a nonlinear process and to evaluate the derivative $\frac{\partial y}{\partial u}$, which represent the output variation against the input one, a mapping neural network is trained online to emulate the process. This is done by back-propagating $E_{net}(k)$ between the process output $y(k)$ and the network output

$y_{net}(k)$. The mapping of the process dynamics can be realized by a two-layer back-propagation network with $(n+m)$ input and one output. The structure chosen for this network should be based on the order of the process. An n^{th} order SISO process, for example, can be approximated by the following linear discrete equation:

$$y_{net}(k) = w_{(n+m+1)} f(\sum_{i=1}^{n} w_i y(k-i) + \sum_{j=1}^{m} w_{n+j} u(k-j) - b_1) - b_2 \qquad (1)$$

with:
w_i and w_{n+j} $(i = 1..n, j = 1..m)$ denote the weights associated to the input layer.
w_{n+m+1} is the weight associated to the output layer.
b_1 and b_2 are the biases associated respectively to the input and the output layer.
f is the activation function of hidden layer. To simplify the calculation, this function is often chosen as the tangent sigmoidal nonlinearity.
If we set:

$$T_1 = \sum_{i=1}^{n} w_i y(k-i) + \sum_{j=1}^{m} w_{n+j} u(k-j) \qquad (2)$$

and with reference to equation (1), the term $\frac{\partial y_{net}}{\partial u}$ can be easily determined by:

$$\frac{\partial y_{net}}{\partial u(k-1)} = w_{(n+m+1)} w_{n+1} (1 - \mathrm{t}\,gsig^2(T_1 - b_1)) \qquad (3)$$

In order to demonstrate the effectiveness of an emulator, we present in this paper the PID Self-Tuning controller as a method for controlling nonlinear systems which requires the emulation of the process dynamics and evaluates the derivative $\frac{\partial y}{\partial u}$ [1, 2, 15, 17].
The structure of the control system using a neural emulator is shown in Figure 1. As shown in this figure, the PID self-tuning controller is implemented using a single-layer network. It consist of a mapping neural network implemented in parallel with the process and a PID controller in the forward path. For this controller, the discrete-time control law can be expressed as follows:

$$u(k) = [w_p \;\; w_i \;\; w_d] \begin{bmatrix} e(k) \\ \sum_{j=1}^{k} e(j) \\ e(k) - e(k-1) \end{bmatrix} = W_{PID} E_{PID} \qquad (4)$$

where W_{PID} the vector of weights w_p, w_i and w_d which represent, respectively, the proportional, integral and derivative gains which should be adjusted. The error signal $e(k)$ is:

$$e(k) = y_c(k) - y(k) \qquad (5)$$

where $y_c(k)$ and $y(k)$ are respectively the reference value and the process output at time k.

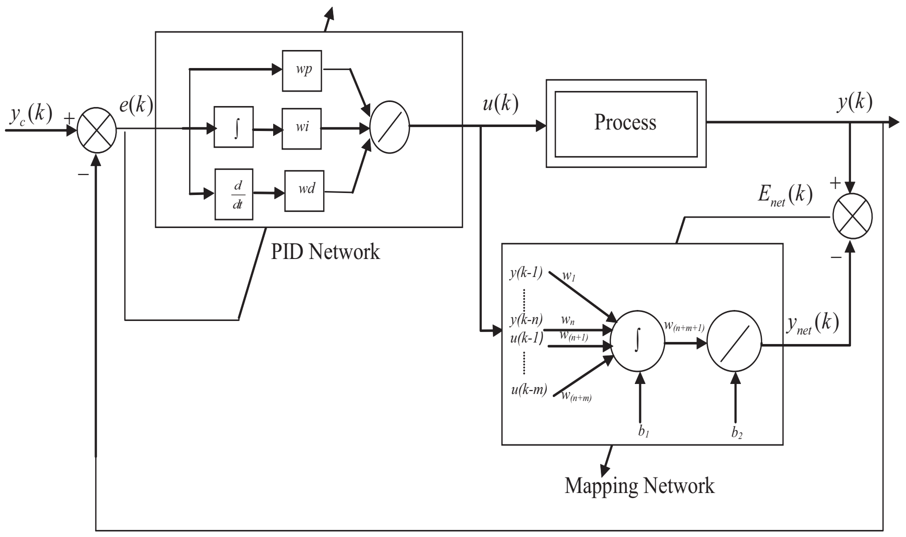

Fig. 1. The structure of control system.

To minimise the control system error by adjusting the gains of the controller, here, gradient descent algorithm is used to minimise the cost function specified as:

$$J = \frac{1}{2}(y_c(k) - y(k))^2 \tag{6}$$

To update the weights of the network which representing the controller gains we need to calculate the gradient term $\frac{\partial J}{\partial W_{PID}}$ which requires the evaluation of $\frac{\partial y}{\partial u}$ [5, 17].

2.1 The effectiveness of the neural emulator

In order to verify the effectiveness of the neural emulator the following nonlinear system is considered [12]:

$$y(k) = \frac{y(k-1)y(k-2)y(k-3)u(k-2)(y(k-3)-1) + u(k-1)}{1 + y^2(k-2) + y^2(k-3)} \tag{7}$$

Since this system is of third order, the input number to the mapping network was selected to be five. The initial weights of the mapping network were random.

However, the initial weights of the PID network were selected as:

$$w_p(0) = 0.1;\ w_i(0) = 0;\ w_d(0) = 0$$

This is to prevent the initial control signal generated to be either too small to actuate the process or too large to drive the system unstable. Learning rates used for the mapping network and the control network were chosen to be $lr = 0.001$ and $lr_{PID} = 0.05$ respectively.

Figures 2 and 3 show respectively the results of the process output and the control signal. It can be seen that the closed loop performances, in this condition, are good.

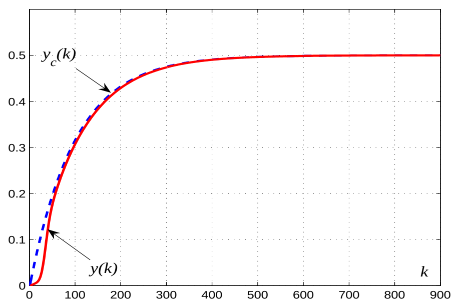

Fig. 2. The evolutions of the desired and the real outputs (Classical Neural Emulator).

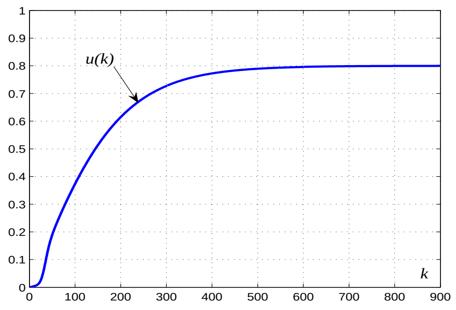

Fig. 3. The control signal (Classical Neural Emulator).

2.2 Limits of the neural emulator

Under certain conditions, the neural emulator may have a relative inability to provide good performance in closed loop. Indeed, the choice of a simple network emulator formed by a single hidden layer and tangent sigmoidal as the activation

function, thanks to its simple derivative, can not be always sufficient in the presence of strong nonlinearities. Furthermore, by examining equation (3), it is obvious that the accuracy of the emulation is closely linked to the initialization and the convergence of the weights of the network emulator. Poor initialization, a slow convergence of the weights (caused by a bad choice of learning rate for example) and the presence of disturbances can lead to poor performance of the control system.
To illustrate this problem, we consider the same system described by the equation (7), using the same learning rate for both networks, the same initialization of the controller gains. The initial weights of the mapping network were random. Between $k = 1600$ and $k = 1700$, a disturbance was injected to the system. Figure 4 illustrates the evolutions of the system output $y(k)$ and of the reference trajectory $y_c(k)$. In presence of this disturbance and with a bad initialization of weights of the mapping network, this figure shows that the performances in regulation and traking are considerably deteriorated.

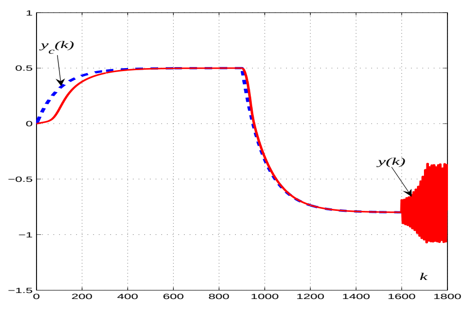

Fig. 4. The evolutions of the desired and the real outputs (Classical Neural Emulator).

Figures 5 and 6 show respectively the results of the control signal and the tuning results of w_p, w_i and w_d.

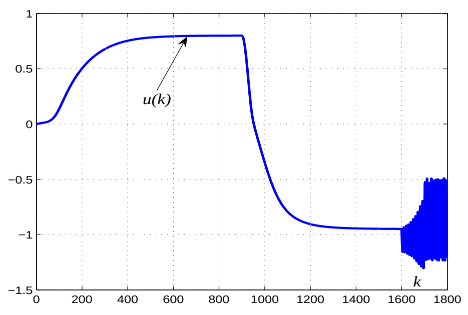

Fig. 5. The control signal (Classical Neural Emulator).

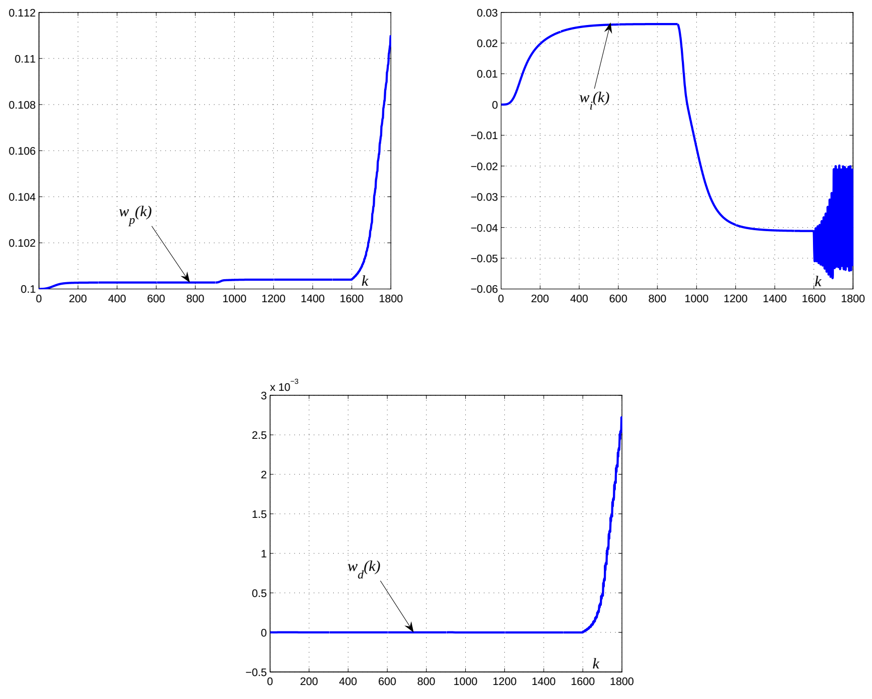

Fig. 6. Evolutions of proportional, integral and derivative gains (Classical Neural Emulator).

3 A Multimodel Emulator For Nonlinear System Controls

3.1 Parametric estimation procedure of an uncoupled multimodel

In this section, the parameter estimation based on the uncoupled state multiple-model approach is presented. The structure of the uncoupled multimodel is given by:

$$\begin{aligned} x_i(k+1) &= A_i x_i(k) + B_i u(k) + D_i \\ y_i(k) &= C_i x_i(k) \end{aligned} \tag{8}$$

where $x_i(k)$ and $y_i(k)$ are, respectively, the state vector and the output for the i^{th} local model, $u(k)$ is the control.
The multimodel output $y_m(k)$ is defined by:

$$y_m(k) = \sum_{i=1}^{N_m} \mu_i(\xi(k)) y_i(k) \tag{9}$$

N_m is the number of local models.
The local model contribution depends on the weighting function $\mu_i(\xi(k))$ that can be obtain from normalized Gaussian function:

$$\mu_i(\xi(k)) = \frac{\exp(-\frac{(\xi(k)-c_i)^2}{\sigma^2})}{\sum\limits_{j=1}^{N_m} \exp(-\frac{(\xi(k)-c_j)^2}{\sigma^2})} \tag{10}$$

c_i is the centre of the i^{th} weighting function and σ is the dispersion for all weighting functions. Decision variable of weighting functions can depend on the measurable state variables and/or input/output variables.
The parametric estimation is based on an iterative minimisation procedure of a quadratic global criterion :

$$J_1 = \frac{1}{2} \sum_{k=1}^{N_H} (y_m(k) - y(k))^2 \tag{11}$$

with Levenberg-Marquardt's algorithm:

$$\theta(it+1) = \theta(it) - \mathit{\Delta}(it)(H(\theta) + \lambda(it)I)^{-1} G(\theta) \tag{12}$$

where N_H is the number of training data, $\theta(it)$ is the vector of the multimodel parameters at a particular iteration it, $\mathit{\Delta}$ is the step size that minimise the criterion in the direction of vector $H^{-1}G$, λ is a scalar and I the identity matrix of appropriate dimension, $H(\theta)$ is the Hessian matrix and $G(\theta)$ is the gradient vector. The calculus of the gradient vector and the Hessian matrix is based on the calculation of sensitivity functions of output multimodel with respect to local models parameters [13, 14].

3.2 The multimodel emulator structure

The resulting local models described by the parameters A_i, B_i, D_i and C_i can be implemented in parallel with the process in different control structure and acting as a plant emulator. This multimodel emulator allows us to obtain the precise dynamics of the process and to easilly calculate the derivative $\frac{\partial y}{\partial u}$ independently of the complexity of the nonlinear dynamics of the process.
The structure of the control system using the proposed emulator is shown in Figure 7.

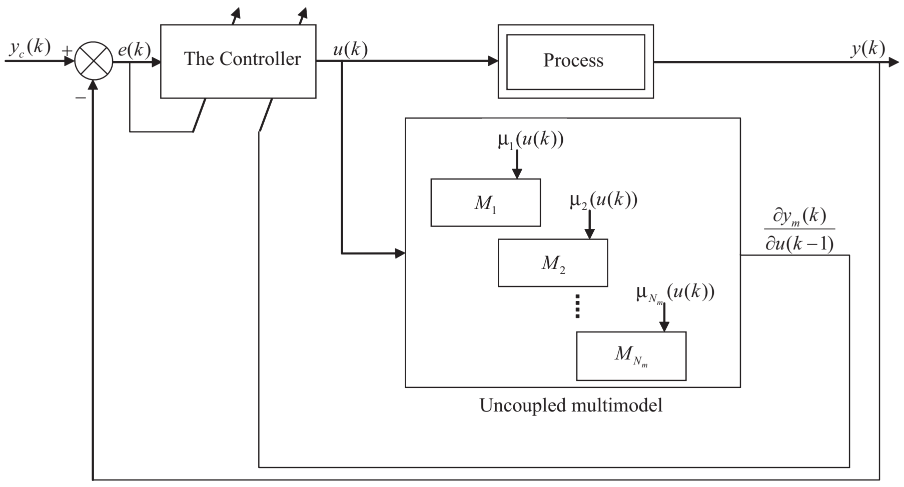

Fig. 7. The structure of control system (multimodel emulator).

The outputs of the partial local models can be defined as follows:

$$\begin{aligned} x_i(k+1) &= A_i x_i(k) + B_i u(k) + D_i \\ y_i(k) &= C_i x_i(k) \end{aligned} \tag{13}$$

where $u(k)$ is the control signal.
The overall output of the emulator is given by:

$$y_m(k) = \sum_{i=1}^{N_m} \mu_i(u(k)) y_i(k) \tag{14}$$

Using equations (13) and (14), the term $\frac{\partial y_m}{\partial u}$ replacing the term $\frac{\partial y}{\partial u}$ can be easily determined from the structure of the multimodel emulator as follows:

$$\frac{\partial y_m}{\partial u(k-1)} = \sum_{i=1}^{N_m} \mu_i(u(k)) \frac{\partial y_i}{\partial u(k-1)} = \sum_{i=1}^{N_m} \mu_i(u(k)) C_i B_i \tag{15}$$

4 Numerical Example

In order to show the considerable contribution in performances of the multimodel emulator, we consider the same nonlinear process described by the equation (7).

4.1 Determination of a models'base

Here, the uncoupled multimodel structure is used. The identification of the multimodel is realized with a global criterion. The input $u(k)$ of the system is formed by the concatenation of piecewise constant signals with variable amplitude ($u(k) \in [-1, 1]$). The multimodel comprises arbitrarily ($N_m = 4$) submodels. The weighting functions μ_i depend on the input signal ($\xi(k) = u(k)$), the centers are: $c_1 = -1$, $c_2 = -0.33$, $c_3 = 0.33$, $c_4 = 1$ and the dispersion $\sigma = 0.4$. A set of 1350 input/output data points is used to build the multimodel. The signal used to validate the uncoupled state multimodel is given by:

$$\begin{cases} u(k) = 0.7\, sin(\frac{2\pi}{250}k) & 1 \leq k \leq 675 \\ u(k) = 0.6\, sin(\frac{2\pi}{250}k) + 0.1\, sin(\frac{2\pi}{250}k) & k > 675 \end{cases}$$

The resulting local models are described by the following expressions:

* Matrice and vectors of state model M_1

$$A_1 = \begin{bmatrix} 0.2641 & 0.3447 \\ -0.0656 & 0.5960 \end{bmatrix} \qquad B_1 = \begin{bmatrix} 0.2468 \\ 0.4988 \end{bmatrix}$$
$$C_1 = [1 \quad 0]$$

* Matrice and vectors of state model M_2

$$A_2 = \begin{bmatrix} -0.0148 & -0.2535 \\ 0.2542 & 0.1197 \end{bmatrix} \qquad B_2 = \begin{bmatrix} 1.1009 \\ 0.6347 \end{bmatrix}$$
$$C_2 = [1 \quad 0]$$

* Matrice and vectors of state model M_3

$$A_3 = \begin{bmatrix} -0.1021 & -0.0434 \\ 0.4000 & 0.8606 \end{bmatrix} \qquad B_3 = \begin{bmatrix} 0.8807 \\ -0.2028 \end{bmatrix}$$
$$C_3 = [1 \quad 0]$$

* Matrice and vectors of state model M_4

$$A_4 = \begin{bmatrix} -0.1987 & 0.0523 \\ 0.6000 & -0.0297 \end{bmatrix} \qquad B_4 = \begin{bmatrix} 0.6794 \\ 0.3698 \end{bmatrix}$$
$$C_4 = [1 \quad 0]$$

The weighting functions are given in Figure 8. The validation results are given in Figure 9. This figure shows the evolutions of the real and the multimodel outputs ($y(k)$ and $y_m(k)$). The simulation results confirm that the method of an uncoupled multimodel identification offers a very satisfactory modeling precision.

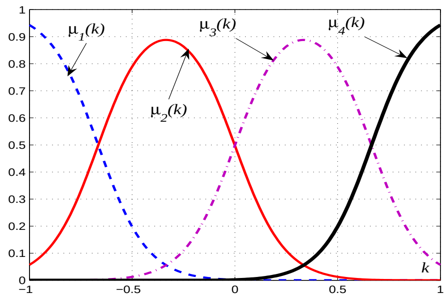

Fig. 8. The weighting functions.

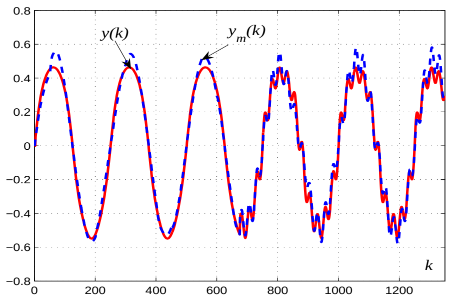

Fig. 9. Nonlinear system output and identified multiple-model output.

4.2 A Multimodel Emulator For PID Self-Tuning controller

The resulting local models is used now to emulate the dynamics of the considered nonlinear system.

Figure 10 and 11 present, respectively, the evolutions of the desired reference trajectory $y_c(k)$ and the system output $y(k)$ in the case of the neural emulator and the multimodel emulator. These figures show that the results recorded in the case of the strategy advanced in this work is far better compared to the classical neural approach.

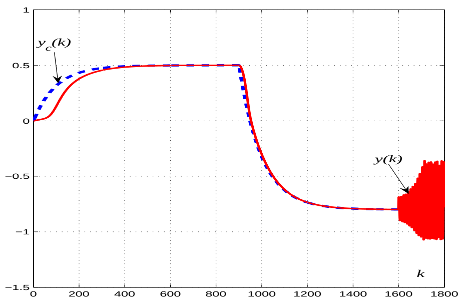

Fig. 10. The evolutions of the desired and the real outputs (Classical Neural Emulator).

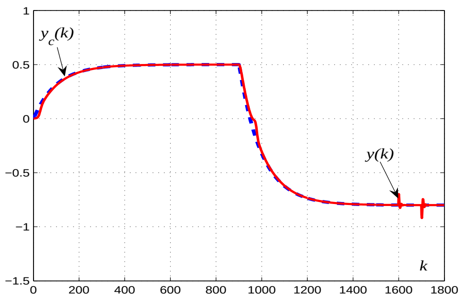

Fig. 11. The evolutions of the desired and the real outputs (Proposed Multimodel Emulator).

Figures 12 and 13 show, respectively, the control signal and the gains w_p, w_i and w_d.

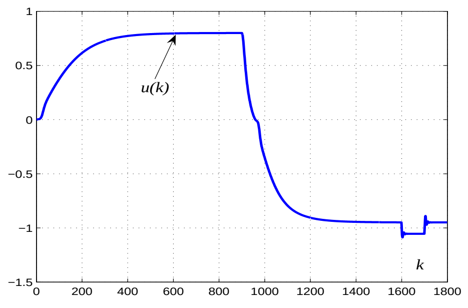

Fig. 12. The control signal (Proposed Multimodel Emulator).

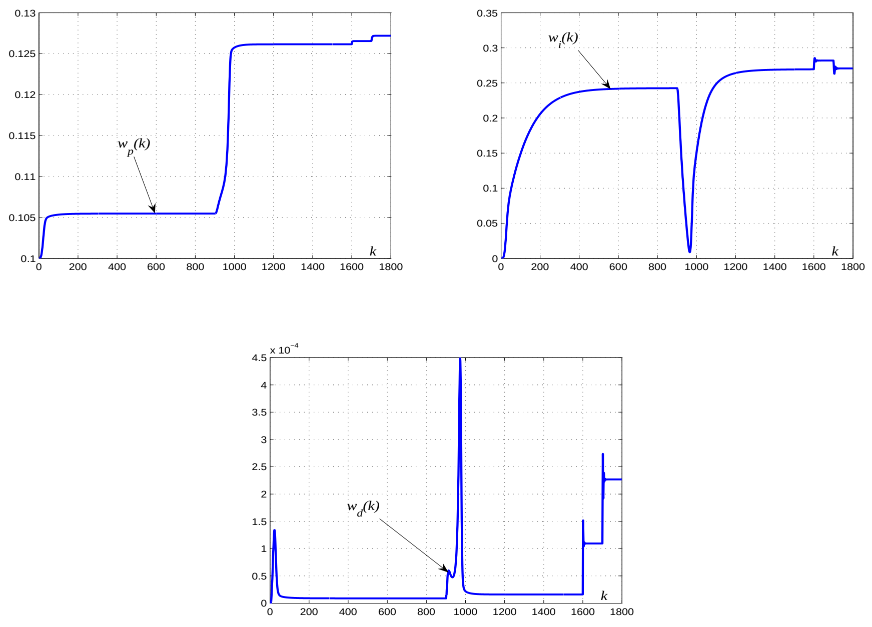

Fig. 13. Evolutions of proportional, integral and derivative (Proposed Multimodel Emulator).

5 Conclusion

In this paper, we have elaborated a multimodel emulator for nonlinear system controls.
This emulator is obtained by using a model's library which result from an off-line identification procedure of nonlinear systems using an uncoupled state multimodel approach. We also compared the performances of this proposed emulator to the classical one based on a neural network. The simulation results show clearly that the proposed multimodel emulator leads to a good closed loop performances relatively to the case where classical neural emulator is applied.

References

1. Akhyar S. and Omatu S., *"Neuromorphic Self-Tuning PID Controller."*, IEEE International Conference on Neural Networks, San Francisco, pp. 552-557, 1993.
2. Bumjin S. and Koivo A.J., *"Neural Adaptive Control of Excavators."*, Proceeding IEEE/RSJ International Conference on Intelligent Robots and Systems, Pittsburgh, PA, pp. 162-167, 1995.
3. Ben Abdennour R., Borne P., Ksouri M. and M'sahli F., *"Identification et commande numérique des procédés industriels."*. Technip, Paris, France, 2001.
4. Filev D., *"Fuzzy modeling of complex systems"*. International Journal of Approximate Reasoning, 5(3), pp. 281-290, 1991
5. Li Q. and Tso S. K., *"PID Tuning Using Neural Networks."*, Proc. World Automation Congress, ISIAC, pp. 137.1-137.6, Albuquerque, USA, 1998.
6. Ltaief M., Abderrahim K., Ben Abdennour R. and Ksouri M., *"Contributions to the multimodel approach: Systematic determination of a models' base and validities estimation "*. International Journal of Automation and Systems Engineering (JASE), 2(3), September 2008.
7. Ltaief M., Abderrahim K., Ben Abdennour R. and Ksouri M., *"A Fuzzy Fusion Strategy for the Multimodel Approach Application "*. WSEAS Trans on Circuits and Systems, 2(4):686-691, 2003.
8. Miller III W.T., Sutton R.S., and Werbos P.J., *"Neural Networks for Control."*, MIT Press, Cambridge, 1990.
9. Murray-Smith R. and Johansen T.A. , *"Multiple Model Approaches to Modeling and Control."*, Taylor & Francis, London, 1997.
10. Messaoud A., Ltaief M. and Ben Abdennour R., *"Supervision Based on Partial Predictors for a Multimodel Generalized Predictive Control: Experimental Validation on a Semi-Batch Reactor "*. International Journal of Modelling, Identification and Control, 6(4):333-340, 2009.
11. Messaoud A., Ltaief M. and Ben Abdennour R., *"Supervision based on a Multipredictor for an Uncoupled State Multimodel Predictive Control"*, The 6th International Conference on Electrical Systems and Automatic Control, JTEA'2010, Hammamet, Tunisia.
12. Narendra K. and Parthasarathy K., *"Identification and control of dynamical systems using neural networks."*, IEEE Transactions on Neural Networks, Vol. 1, No. 1, pp. 4-27, 1990.
13. Orjuela R., Maquin D. and Ragot J., *"Nonlinear system identification using uncoupled state multiple-model approach"*. In Workshop on Advanced Control and Diagnosis, ACD'2006, Nancy, France.

14. Orjuela R., *"Contribution à l'estimation d'état et au diagnostic des systèmes représentés par des multimodèles."*, Thèse de l'Institut Nationale Polytechnique de Lorraine, Novembre 2008.
15. Swiniarski R.W., *"Novel Neural Network Based Self-Tuning PID Controller Which Uses Pattern Recognition Technique."*, American Control Conference, San Diego, pp. 3023-3024, 1990.
16. Takagi T. and Sugeno M., *"Fuzzy identification of systems and its applications to model and control"*. IEEE Transactions on Systems, Man, and Cybernetics, 15 :116-132, 1985.
17. Talmoudi S., Ben Abdennour R., Abderrahim K. and Ksouri M., *"Multi-modèle et multi-commande neuronaux pour la conduite numérique des systèmes non linéaires et non stationnaires."*, Conférence Internationale Francophone d'Automatique, Nantes, 2002.
18. Talmoudi S., Ben Abdennour R. , Abderrahim K. and Ksouri M., *"A New technique of validities'computation for multimodel approach "*. Wseas Transactions on circuits and systems, 2(4):680-685, Octobre 2003.
19. Talmoudi S., Abderrahim K., Ksouri M. and Ben Abdennour R., *"Multimodel approach using neural networks for complex systems modeling and identification "*. Journal of Nonlinear Dynamics and Systems, 8(3), 2008.