Modeling of a Turbocharger Diesel Engine using Neural Network

X.Dovifaaz, M. Ouladsine A. Rachid

Automatic Systems Laboratory
7, Rue du Moulin Neuf - 80000 Amiens
FRANCE
Tel.: (+33) 3 22 82 76 63 - Fax : + 33 3 22 82 76 82

Abstract : This paper describes a neural modeling of a thermodynamic system. It presents the training of neural networks, based on the theory of backpropagation, like the Levenberg-Marquardt method, to model a turbocharger diesel engine. The model contains several static and dynamic neural blocks, corresponding to the equations of the engine. The paper describes the algorithms of learning and the application to the real system.

Key Words: *neural-network models, backpropagation learning, Levenberg-Marquardt method, diesel engines*.

1 Introduction

Neural networks are subjected to continuous researches during the last years. For instance, they are used for modeling [9][4][7], control, vision and diagnosis [11]. They bring many non negligible important benefits, compared with classic ones, whose theoretic developments are not always easy, in front of complex systems. Among the advantages, neural networks can describe or control nonlinear complex systems with precision, due to the fact that they include themselves nonlinearities in their structure. Moreover, using them sometimes allows avoiding difficult mathematics analysis, thanks to their simple structure (nodes and connections). Obviously, they need to be theoretically studied. Especially, theoretical analysis is done to study their stability for controlling linear or nonlinear systems [13]. One benefit of neural networks he theory is the same for a large class of systems, while Lyapounov theory need for instance to consider the class of the system. In this paper, we applied neural techniques to a turbocharger engine. Automotive must indeed strict constraints concerning the exhaust emissions. Our objective is to find an optimal control minimizing pollution and consumption. But the equations are complex and non-linear [1]. This explains why artificial neural networks are used in modeling. The present paper only focuses on modeling of the system.

It is presented as follows: section 1 presents the theory of neural network for modeling the behavior of the engine. Section 2 describes the neural optimal control theory used in our application, with a first part concerning the learning algorithms, and the second, the structure identification of neural networks. Section 3 shows the application of neural modeling to the diesel engine, which includes some modifications in the learning procedures. The first part of this section describes the diesel engine operating, while the second focuses on the model construction of the engine.

2 Neural Network Modeling

Among the advantages of the neural networks, is their capacity to model the behavior of nonlinear system. This is due to the fact that their structure contains nonlinear simple elements: the nodes and their nonlinear activation function. Neural modeling consists in finding an optimal neural network describing the system, this means the value of the parameters, and the structure of the network. Some learning algorithms, whose aim is to make the output estimation of the network equal to the corresponding state of the real system, calculate the parameters or weights. The structure is selected by studying the network that gives the best results after learning.

2.1 Neural network learning

2.1.1 Static networks

The learning consist in calculating the weights such that the estimated output is closed to the corresponding real state of the system.
To satisfy this objective, learning algorithms minimizing the following quadratic criterion are used:

$$E(W) = \frac{1}{2.\mathrm{N}}.\sum_{k=1}^{N}\left(e(k,W)\right)^2$$

$$e(k,W) = e(k) = y(k) - \hat{y}(k,W)$$

Thus, the error between the estimation and the real state is minimized. The simplest method is the backpropagation learning [9][10][3]. This algorithm brought to other more complexes but efficient methods that converge more rapidly. The updating rules are as follows:

$$W^{i+1} = W^i - m.\left[R(W^i)\right]^{-1}.\nabla E(W^i) \qquad (1)$$

where W^i is the parameter vector after i updates, $\nabla E(W^i)$ is the gradient of the quadratic criterion to minimize, defined as :

$$\nabla E(W) = \begin{pmatrix} \frac{\partial E}{\partial w_1} \\ \vdots \\ \frac{\partial E}{\partial w_n} \end{pmatrix} \qquad (2)$$

with:

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i}\left(\frac{1}{N}.\sum_{k=1}^{N}\left(e(k)\right)^2\right) = \frac{1}{N}.\sum_{k=1}^{N}\left(\frac{\partial e(k)}{\partial w_i}.e(k)\right) = -\frac{1}{N}.\sum_{k=1}^{N}\left(\frac{\partial \hat{y}(k)}{\partial w_i}.e(k)\right) \qquad (3)$$

thus, we have :

$$\nabla E(W) = -\frac{1}{N}.\Psi^T.e \qquad (4)$$

where :

$$\Psi = \begin{bmatrix} \frac{\partial e(1)}{\partial w_1} & \cdots & \frac{\partial e(1)}{\partial w_n} \\ \vdots & & \vdots \\ \frac{\partial e(N)}{\partial w_1} & \cdots & \frac{\partial e(N)}{\partial w_n} \end{bmatrix}; \quad e = \begin{bmatrix} e(1) \\ \vdots \\ e(N) \end{bmatrix}$$

In (1) the term R appears. It depends on the learning algorithms. Three of them are describe here:

- **The gradient backpropagation** :

R=I, identity matrix.
The gradient backpropagation is rarely efficient. Choosing the identity matrix is incompatible with the different scales between the inputs and the outputs. The updating rule comes from the development of the criterion at the first order :

$$E(W^{i+1}) = E(W^i + \Delta W) = E(W^i) + \left[\nabla E(W)\right]^T_{W=W^i}.\Delta W + o\|\Delta W\|$$

where W^i is the parameter vector after i update
thus

$$\Delta E = E(W^i + \Delta W) - E(W^i) \approx \left[\nabla E(W)\right]^T_{W=W^i}.\Delta W$$

Minimizing E is the same as minimizing ΔE and then we obtain :

$$\Delta W = -m.\left[\nabla E(W)\right]_{W=W^i} \qquad (5)$$

- **the Gauss-Newton algorithm**:

It comes from the development of the criterion E to the second order :

$$\Delta E = \left[\nabla E(W)\right]^T_{W=W^i}.\Delta W + \frac{1}{2}.\Delta W^T.H(W^i).\Delta W + o\|\Delta W\|^2$$

or,

$$E(W^{i+1}) = E(W^i + \Delta W) = E(W^i) + \left[\frac{\partial E}{\partial W}\right]^T_{W=W^i}.\Delta W + \frac{1}{2}.\Delta W^T.\left[\frac{\partial^2 E}{\partial W \partial W^T}\right]_{W=W^i}.\Delta W + o\|\Delta W\|^2 \qquad (6)$$

By searching the variation of the parameter vector ΔW that brings the greater decrease of the criterion, we can write :

$$\left[\frac{\partial \Delta E}{\partial \Delta W}\right]_{W=W^i} = 0 \approx H(W^i).\Delta W + \left[\nabla E(W)\right]_{W=W^i}$$

The corresponding algorithm is the Newton algorithm. To permit convergence even for points far from the minimum, the term m is added, and then we have :

$$\Delta W = -m.H(W^i)^{-1}.\left[\nabla E(W)\right]_{W=W^i}$$

In the case of the Gauss-Newton algorithm, R is an approximation of the criterion's hessian, whose elements are presented in (9) :

$$\frac{\P E}{\P w_i} = \frac{1}{N}.\sum_{k=1}^{N}\frac{\P e(k)}{\P w_i}.e(k)$$

$$\frac{\P^2 E}{\P w_j w_i} = \frac{1}{N}.\sum_{k=1}^{N}\left(\frac{\P e(k)}{\P w_j}.\frac{\P e(k)}{\P w_i} + e.\frac{\P^2 e(k)}{\P w_j \P w_i}\right) \quad (7)$$

e is a residual term. By neglecting the second order terms, we have :

$$\frac{\P^2 E}{\P w_j w_i} \approx \frac{1}{N}.\sum_{k=1}^{N}\left(\frac{\P e(k)}{\P w_j}.\frac{\P e(k)}{\P w_i}\right) \quad (8)$$

or, under a matrix form :

$$R = \frac{\P^2 E}{\P w_j w_i} = H = \frac{1}{N}.\Psi^T.\Psi \quad (9)$$

If this algorithm is more efficient than the precedent one, some problems like the singularity of the matrix R. The following method can avoid this problem.

- **The Levenberg-Marquardt algorithm** :

R=H+β.I; H is defined in (9). β is a scalar.
The Levenberg-Marquardt algorithm [6][3][10] avoid singularities, and permits a more rapid convergence of the parameters. When β=0, we obtain the Gauss-Newton algorithm, and when β goes towards ∞, it tends to the gradient propagation. In our application, we use the including the following rate, as presented in [8] :

$$r^{(i)} = \frac{E(W^{(i)}) - E(W^{(i)} + \Delta W^{(i)})}{E(W^{(i)}) - L(W^{(i)} + \Delta W^{(i)})} \quad (10)$$

where :

$$L(W+\Delta W) = \frac{1}{N}\sum_{k=1}^{N}\left(y(k) - \hat{y}(W,k) - \Delta W^T.\frac{\P \hat{y}}{\P W}\right)^2$$

$$= E(W) + \Delta W^T.\Psi(k,W) + \frac{1}{2}.\Delta W^T.H.\Delta W$$

- If r is near 1, it means that we are near a minimum. Then β is decreased to approach the Gauss-Newton direction.
- If r is near 0, parameters are far from the optimum. β is increased, and the direction search is the gradient one.

2.1.2 Dynamic neural networks learning

The learning is different for prediction or simulation models. In the first case, the delayed inputs of the dynamical network are independent of the weights, and can be considered as external inputs. In the second case, the delayed inputs depend on the weights. Figure 1 shows the dynamic neural learning scheme for a simulation model.

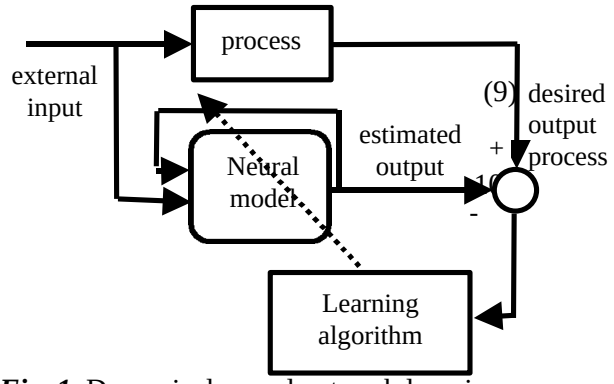


Fig. 1. Dynamical neural network learning.

The updating rule includes now the derivative terms:

$$\frac{\P E}{\P w_i} = -\frac{1}{N}.\sum_{k=1}^{N}\left(\frac{\Delta \hat{y}(k)}{\Delta w_i}.e(k)\right)$$

with

$$\frac{\Delta \hat{y}(k)}{\Delta w_i} = \frac{\P \hat{y}(k)}{\P w_i} + \sum_{i=1}^{n_y}\left(\frac{\P \hat{y}(k)}{\P \hat{y}(k-i)}.\frac{\P \hat{y}(k-i)}{\P w_i}\right) \quad (11)$$

ny is the order of the model.

2.2 Neural network structure identification

A dynamical network is presented in figure 2. It contains layers, connections, and activation functions that are to be fixed. For our application, full-connected networks are used.

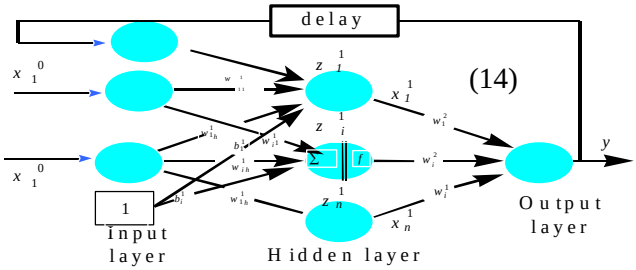


Fig. 2. Recurrent neural network structure.

Several activation functions can be employed. Networks containing nodes with radial basis activation functions give useful responses on a restricted domain of its input. They seem to be efficient for classification problems. Another frequently activation used for modeling is the sigmoid function :

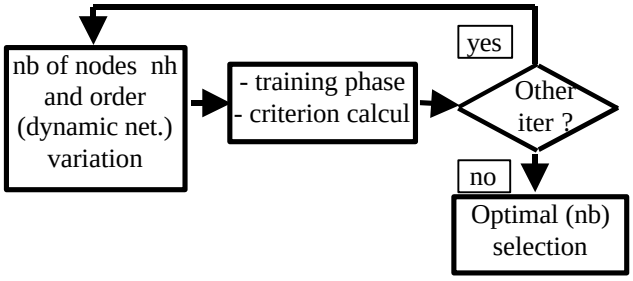
$$f_i = \frac{1}{1+e^{x_i/T}}$$

x_i is the input of the node i, and T one parameter.

Some authors like [12] vary the parameter T, observing some improvements. But it is generally considered as a useless additive parameter that is added to the others. We used the sigmoid function for our application, with the value T=1. To find the structure, several networks are trained, with different order and different node number. The network that gives the best criterion on test and validation data is selected, as shown below :

Fig. 3. Structure identification.

The loop consists in training the network for several numbers of nodes, *nh*, and orders. The network, which gives the smaller criterion on the training and validation data, is then selected.



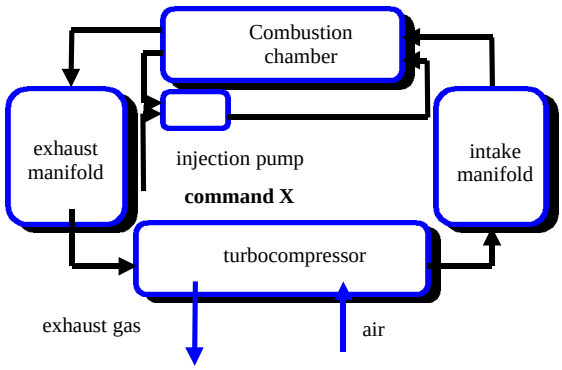
3 Application to the Engine

3.1 Description of the diesel engine

The engine used in our application is a BMW's one. It can be decomposed into subsystems as presented in diagram (4). The atmospheric air goes through the compressor, the air intake manifold, and the combustion chamber.

Fig.4.Turbocharged diesel engine plant.

The injection pump injects fuel in the combustion chamber while the valves are closed, and the mixture burns. The gases produced by the explosion passe through the exhaust manifold and the turbine and goes outside. Two states have been modeled : the engine speed, *R*, and pressure in the intake manifold, *P*. The only command that we consider is the position of the accelerator, *X*.



Modeling of a Diesel engine have partly been investigated by many authors [5][2]. The acknowledge of some relations allows to structure the neural model more easily. After some simplification, we obtain the equations :

$$p(k+1) = F_p\begin{pmatrix} p(k),...,p(k-n_{p1}), \\ w(k),...,w(k-n_{w1}) \end{pmatrix}$$

$$w(k+1) = F_w\begin{pmatrix} w(k),...,w(k-n_{w2}), \\ p(k),...,p(k-n_{p2}), \\ X(k),...,X(k-n_x) \end{pmatrix}$$

$$\dot{m}_f(k+1) = f_{debf}\ \left(w(k+1), X(k+1)\right)$$

$$\dot{m}_c(k+1) = f_{deba}\ \left(w(k+1), p(k+1)\right)$$

$$Opac(k+d) = f_{opac}\ \begin{pmatrix} op(k+d-1),...,op(k+d-n_{op}), \\ deba(k), debf(k), w(k) \end{pmatrix} \quad (12)$$

X is the acceleration, W the engine speed, P the inlet collector pressure, $\dot{m}_f$ the fuel flow, $\dot{m}_c$ the inlet air flow, and Opac the opacity of the exhaust gas.

3.2 Diesel engine neural modeling

3.2.1 Model structure identification

Each equation corresponds to a neural network model. Static networks model static equations, while dynamics are modeled by dynamic network. We applied the method of figure (3), with the Levenberg-Marquardt method. The following model structure in figure (5) was obtained. 5 neural blocks with one hidden layer, compose the complete model to estimate the 5 states.

The curves in figure (6) present the criterion evolving according to the structure of each network. It includes the dynamic model orders (equation 12) and the number of hidden nodes *nh* of each network. The optimal structure is displayed for each curve.

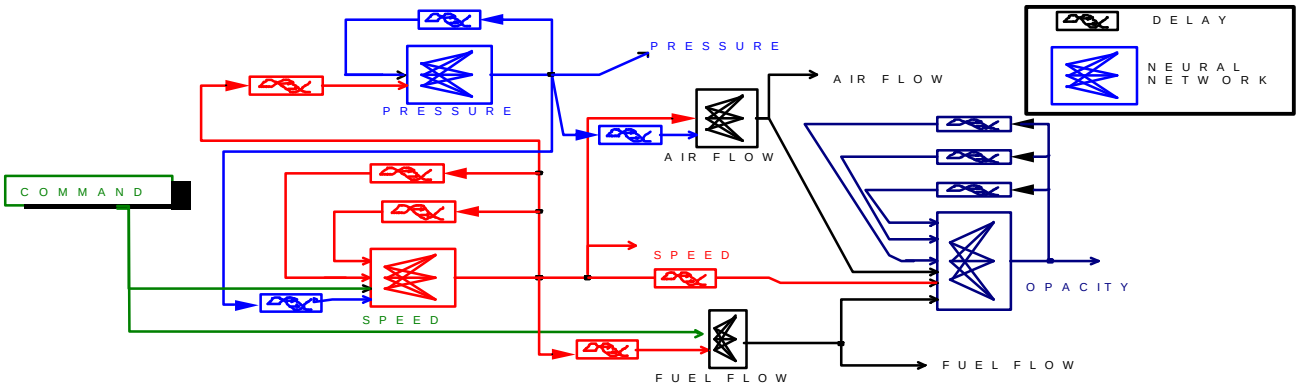


Fig. 5. Neural model of the engine.

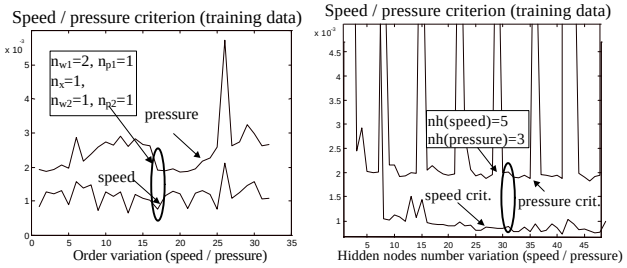


Fig. 6.Speed / pressure criterion

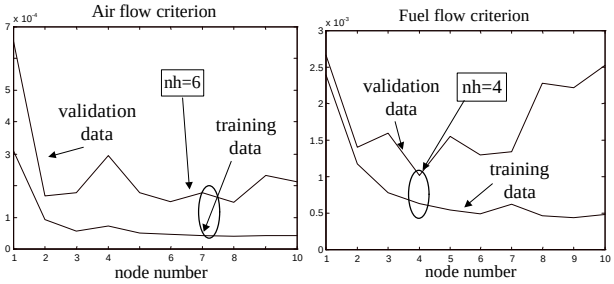


Fig. 7. Air and fuel flow criterion.

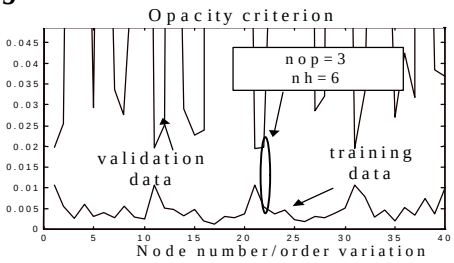


Fig. 8. Opacity criterion.

The structure identification of the speed and pressure models was modified. Indeed, the two neural blocks have been learned independently. But after connecting them for simulation, they can give bad results. The following figure (9) presents simulation results after independent training, for a given structure. While the simulation of each separated block presents good results, the complete one, including the two states, leads to unsatisfying results. The divergence can be explained by the amplification of small errors on the pressure estimation, entering in the speed model.

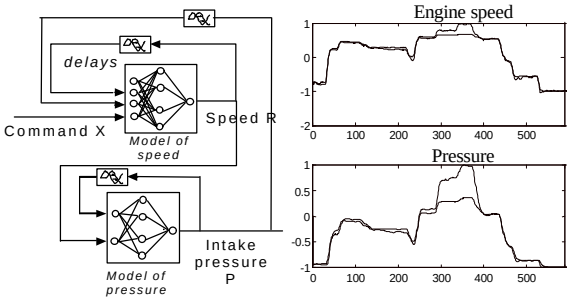


Fig. 9. Speed / pressure independent training simulation.

The procedure in figure (10) describes the modified training: the two networks are successively trained. The estimated resulting speed is used to learn the pressure model :

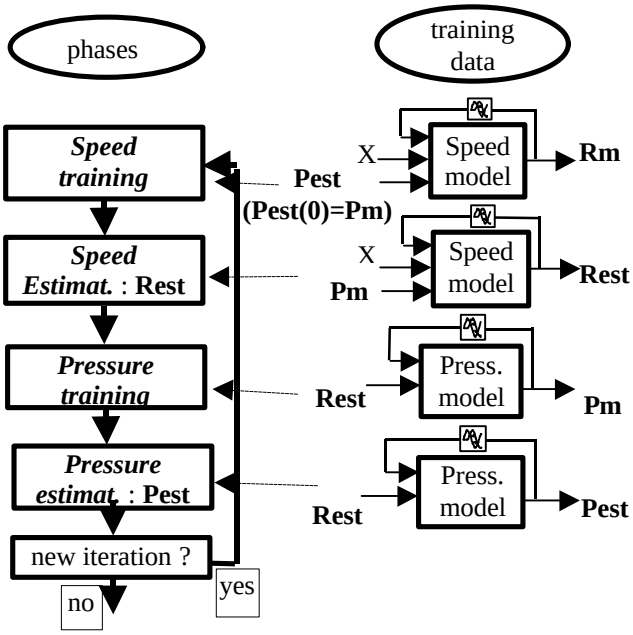


Fig. 10. Training procedure for speed and pressure model.

Pest / Pm :the last estimated / measured pressure vector
Rest / Rm :the last estimated / measured speed vector

The calculated criterion is displayed on figure 6 for the training and validation data, in accordance with the order (on left) and the node number (on right).

The criterion is obviously higher for validation data, only used for simulation, and not in training.

3.2.2 Model simulation

Displayed results displayed below are issued from the complete identified simulation model, with just the acceleration as input

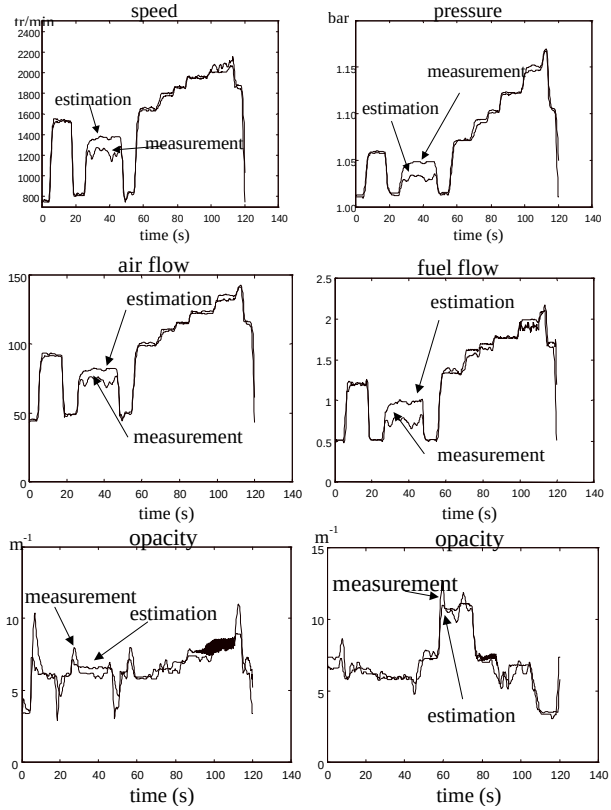


Fig. 11. Simulation results.

The precedent results have been obtained with only the acceleration as global input of the entire model. They are thus satisfying. Other models can be tested, without considering blocks like it was done. Physically, all the states depend on only one command, that is the acceleration. Thus, each state can be considered as an output of a black box, that means a neural network, whose input is the acceleration signal. The disadvantage would be that this needs bigger neural networks.

4 Conclusions

This work is realized in the DIVA project (Diagnostic des Véhicules Avancés-France-), on the control optimization of exhaust emissions of a diesel engine. In this article, we presented a global neural model of our engine. This operation has been realized with the Levenberg-Marquardt algorithm, which appeared to be satisfying. Few acknowledge on our system were needed; this characterizes the flexibility of using neural network for different systems. A comparison can be done in the further between the presented results and those that would come from another model structured, as proposed in last section. Our neural model of the engine will be useful for testing some neural control of the engine.

References :

[1] Blanke M. and J.S.Andersen (1984). *On modeling large two stroke diesel engines : News results from identification*. *IFAC, 9th. Triennal World Congress*. Budapest, Hungary. 2015-2020.

[2] Blanke M (1986). *Requirements of adaptive techniques for enhanced control of large Diesel engines*. IFAC. Lund, Sweden. 197-202.

[3] Bloch G., P.Thomas, M.Ouladsine and M.Lairi (1996). *On several Outlier robust Training Rules for identification of nonlinear systems*. 8th Int.Conf. on Neural Networks and their applications.NEURAL'96,Marseille, 20-23.

[4] Chen S., S.A.Billings, and P.M.Grant (1990). *Non-linear system identification using neural networks*. Int.J.Control. **51**. 1191-1214.

[5] Cook J.A., J.W.Grizzle and J.Sun (1996). *Engine control syst.* The control Handbook. 1261-1274.

[6] Ljung L. (1983). In: *Theory and Practice of Recursive Identification* (Torsten Söderström).

[7] Narendra K.S. and K.Parthasarathy (1990). *Ident. and cont. of dyn. Syst. using neural networks*. IEEE Trans. On Neural Networks. **1**,4-27.

[8] Noorgaard M., O.Ravn, L.K.Hansen, N.K.Poulsen (1996). *The NNSYSID Toolbox. A Matlab Toolbox for Syst.Ident.with Neural Networks*. Proc. of the 1996 IEEE Int.Symp. on Comp. Aided Cont.Syst.Design. Dearborn, Michigan, USA. pp.374-379.

[9] Pham D.T. (1995). In: *Neural Networks for Identification, Prediction and Control* (Liu X). Springer-Verlag London Limited.2nd edition.

[10] Söberg J., H. Hjalmarsson, and L.Ljung (1995). *Nonlinear Black-box Modeling in System Identification : a Unified Overview*. Automatica. **31**. n°12. pp.1691-1724.

[11] Sorsa T. and H.N.Koivo (1993). *Application of artificial neural networks in process fault diagnosis*. Automatica. **29**. 4. 843-849.

[12] Jodouin, J.F. (1993). *Rés. de neurones et trait. du langage naturel. Etude des rés. de neurones récurrents et de leurs représentations*, Notes et doc. LIMSI/CNRS, 93-16.

[13] Renders J-M (1995*). Algorithmes génétiques et réseaux de neurones*. Hermes ed. 229-2