IMPLEMENTING THE LEVENBERG-MARQUARDT ALGORITHM ON-LINE: A SLIDING WINDOW APPROACH WITH EARLY STOPPING

Fernando Morgado Dias[1], Ana Antunes[1], José Vieira[2], Alexandre Manuel Mota[3]

[1]*Escola Superior de Tecnologia de Setúbal do Instituto Politécnico de Setúbal, Departamento de Engenharia Electrotécnica, Campus do IPS, Estefanilha, 2914-508 Setúbal, Portugal Tel: +351 265 790000, Email:{ fmdias,aantunes}@est.ips.pt*
[2]*Escola Superior de Tecnologia de Castelo Branco, Departamento de Engenharia Electrotécnica, Av. Empresário, 6000 Castelo Branco, Portugal, Tel: +351 272 330300, Email: zevieira@est.ipcb.pt*
[3]*Departamento de Electrónica e Telecomunicações, Universidade de Aveiro, 3810 - 193 Aveiro, Portugal, Tel: +351 234 370383, Email: alex@det.ua.pt*

Abstract: The Levenberg-Marquardt algorithm is considered as the most effective one for training Artificial Neural Networks but its computational complexity and the difficulty to compute the trust region have made it very difficult to develop a true iterative version to use in on-line training. The algorithm is frequently used for off-line training in batch versions although some attempts have been made to implement iterative versions.
To overcome the difficulties in implementing the iterative version, a batch sliding window with Early Stopping version, which uses a hybrid Direct/Specialized evaluation procedure is proposed and tested with a real system.

Keywords: Artificial Neural Networks, Levenberg-Marquardt Algorithm, On-line Training, Direct Training and Specialized Training.

1. INTRODUCTION

Among the many algorithms used for training in the field of Artificial Neural Networks (ANNs) the Levenberg-Marquardt has been considered as the most effective one, but its use has been mostly restricted to the off-line training because of the difficulties to implement a true iterative version. These difficulties come from computing the derivatives for the Hessian matrix, inverting this matrix and computing the region for which the approximation contained in the calculation of the Hessian matrix is valid (the trust region).
In the present work a different approach is suggested: the use of the Levenberg-Marquardt algorithm on-line through a batch version with sliding window and Early Stopping. This way the Levenberg-Marquardt algorithm can be applied as in the off-line approaches. The implementation also uses a hybrid Direct/Specialized evaluation procedure and the whole set is tested in a real system composed of a reduced scale prototype kiln.

2. THE NEWTON, GAUSS- NEWTON AND LEVENBERG-MARQUARDT ALGORITHMS

Starting from the Taylor series approach of second order, for a generic function F(x), the following can be written:

$$F(x_{k+1}) = F(x_k + \Delta x_k) \simeq F(x_k) + G(x,k).\Delta x_k + \tfrac{1}{2}.\Delta x_k.H(x,k).\Delta x_k \quad (1)$$

where G(x,k) is the gradient of F(x), Δx_k is x_{k+1}-x_k and H(x,k) is the Hessian matrix of F(x).
If the derivative of equation 1 in respect to Δx_k is taken, equation (2) will be obtained:

$$G(x,k) + H(x,k).\Delta x_k = 0 \quad (2)$$

This equation can be re-written in the following form:

$$\Delta x_k = -H(x,k)^{-1}.G(x,k) \quad (3)$$

The updating rule for the Newton algorithm is then obtained:

$$x_{k+1} = x_k - H(x,k)^{-1}.G(x,k) \quad (4)$$

Considering a generic quadratic function as the objective function to be minimized, as represented in equation 5 for a Multi Input Multi Output system (here the iteration index is omitted and i is the index of the outputs):

$$F(x) = \sum_{i=1}^{N} v_i^2(x) \quad (5)$$

The gradient can be expressed in the following form:

$$G(x) = 2J^T(x).v(x) \quad (6)$$

and the Hessian matrix can be expressed in the following form:

$$H(x) = 2J^T(x).J(x) + 2S(x) \quad (7)$$

where J(x) is the Jacobian and S(x) is:

$$S(x) = \sum_{i=1}^{N} v_i(x).\frac{\partial^2 v_i(x)}{\partial x_k \partial x_j} \quad (8)$$

if it can be assumed that S(x) is small when compared to the product of the Jacobian, then the Hessian matrix can be approximated by the following:

$$H(x) \simeq 2J^T(x).J(x) \quad (9)$$

This approach gives the Gauss-Newton algorithm:

$$\triangle x_k = -[2J^T(x_k).J(x_k)]^{-1}.2J^T(x_k).v(x_k) \quad (10)$$

One limitation that can happen with this algorithm is that the simplified Hessian matrix might not be invertible. To overcome this problem a modified Hessian matrix can be used:

$$Hm(x) = H(x) + \mu I \quad (11)$$

where I is the identity matrix and μ is a value such that makes Hm(x) positive definite and therefore can be invertible.
This last change in the Hessian matrix corresponds to the Levenberg-Marquardt algorithm:

$$\triangle x_k = -[2J^T(x_k).J(x_k) + \mu_k I]^{-1}.2J^T(x_k).v(x_k) \quad (12)$$

where μ is now written as μ_k to show that this value can change during the execution of the algorithm.
The Levenberg-Marquardt algorithm is due to the independent work of both authors in (Levenberg, 1944) and (Marquardt, 1963).
The selection of μ is of extreme importance for the functioning of the algorithm since it is responsible for stability (when assuring that the Hessian can be inverted) and the speed of convergence. It is therefore worth for a closer look of how to calculate this value.
The modification of the Hessian matrix will only be valid in a neighbourhood of the current iteration. This corresponds to search for the correct update for the next iteration x_{k+1} but restricting this search to:

$$|x-x_k|\}| \leq \delta_k \quad (13)$$

There is a relationship between δ_k and μ_k since raising μ_k makes the neighbourhood δ_k diminish (Norgaard et al., 2000).
As an exact expression to relate these two parameters is not available, many solutions have been developed (Norgaard et al., 2000).
The one used in the present work was proposed by Fletcher (Norgaard et al., 2000) and uses the following expression:

$$r_k = \frac{V_N(x_k) - V_N(x_k + f_k)}{V_N(x_k) - L_k(x_k + f_k)} \quad (14)$$

to obtain a measure of the quality of the approximation. Here V_N is the function to be minimized and L_k is the estimate of that value calculated from the Taylor series of second order and f_k is the search direction, in the present situation, the search direction given by the Levenberg-Marquardt algorithm.
The value of r_k is used in the determination of μ_k according to the following algorithm:

1-Choose the initial values of x_0 e μ_0.
2-Calculate the search direction f_k.
3-If $r_k > 0.75$ then set $\mu_k = \mu_k/2$.
4-If $r_k < 0.25$ then set $\mu_k = 2.\mu_k$.
5-If $V_N(x_k+f_k) < V_N(x_k)$ then the new iteration is accepted.
6-If the stopping condition is not met, return to step 2.

3. ON-LINE IMPLEMENTATIONS

Some attempts have been made to build on-line implementations for the Levenberg-Marquardt algorithm. The difficulties, as pointed out before, come from computing the derivatives for the Hessian matrix, inverting this matrix and computing the trust region, the region for which the approximation contained in the calculation of the Hessian matrix is valid.
Among the attempts that can be found in the literature, it is worth to note the work done by Ngia (Ngia, 2000) developing a modified iterative Levenberg-Marquardt algorithm which includes the calculation of the trust region and the work in (Ferreira et al., 2002) which implements a Levenberg-Marquardt algorithm in sliding window mode for Radial Basis Functions.

3.1 Sliding window with Early Stopping

For the present work the on-line version of the Levenberg-Marquardt algorithm was implemented using a sliding window with Early Stopping and static test set for evaluation purposes which was collected in advance.
The Early Stopping (Morgan et al., 1990), (Sjöberg, 1995) technique is used for avoiding the overfitting problem and was preferred for the present work since it has less computational burden.
Other techniques that could have been used were Regularization and Prunning techniques.
A technique to avoid overtraining is absolutely necessary when systems that are subject to noise are considered. After the initial phase of training the model will start to learn information from the noise present in the training set, or if the data is filtered, not all the information the model might learn will be valid.
The use of Early Stopping is nevertheless not as direct as in off-line application since the training set is changing from one iteration to the next one, as it is illustrated in figure 1, where the two situations that occur in the training set are represented, with m representing the iteration index.

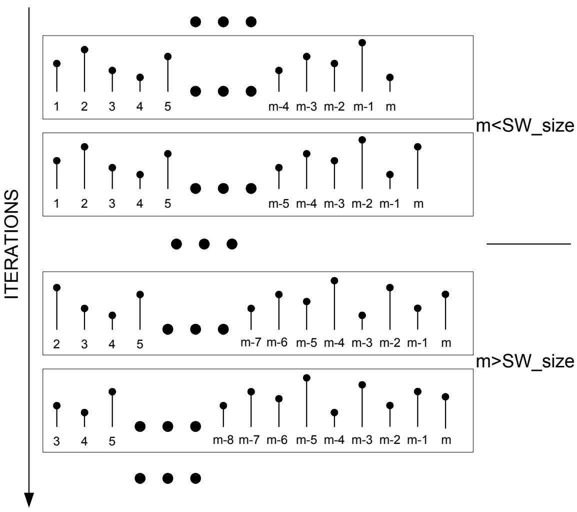

Figure 1 – Detail of the operation of the sliding window.

The training starts after some data has been collected but usually before the amount of data has reached the pre-defined size for the sliding window. This is done to save some of the time necessary to complete collecting the points. In this phase the sliding window is growing, but not one sample for each training epoch, since all of the sampling time is used for training. This way in each sampling period several training epochs are performed.
In the second phase, after the number of points collected exceeds the size of the sliding window, the sliding window changes due to the entrance of new samples and due to the removal of the same amount of samples for each sampling period.
This change in the sliding window implies that in some cases the value of the test error "jumps" from one iteration to the next one and that could be erroneously interpreted as overtraining. This obliges extra care at the interpretation that leads to the verification of the overtraining situation.

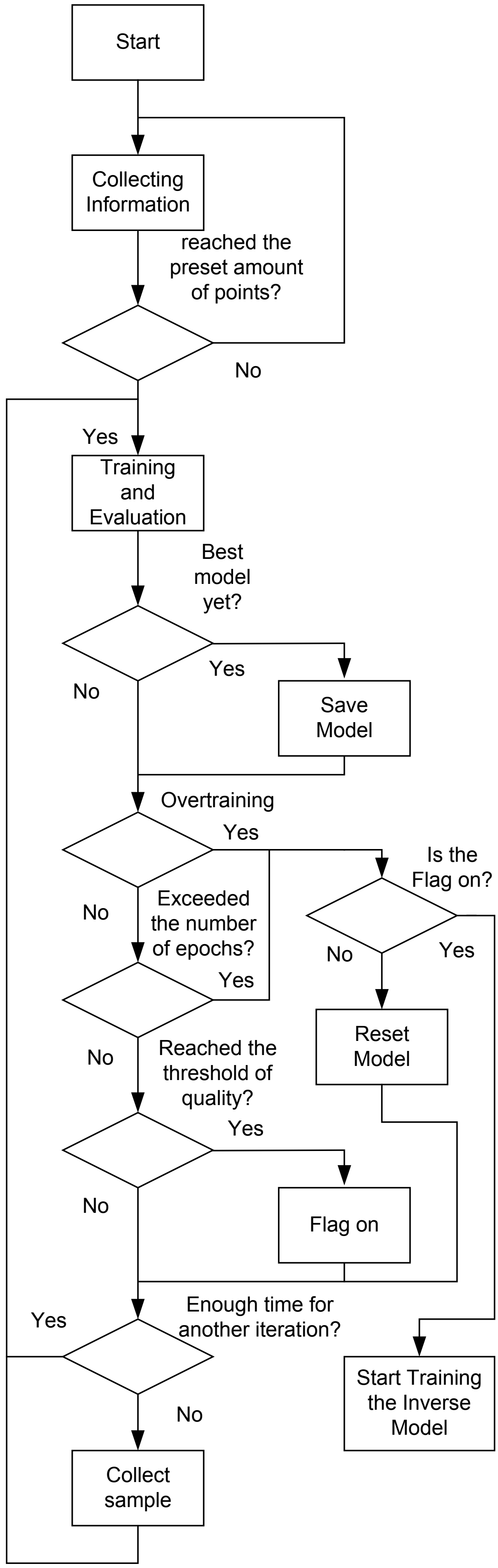

Figure 2 – Procedure for identification of a direct model on-line.

The procedure used for the identification of the direct model on-line is represented in figure 2.

As was already explained, training starts when a predefined amount of points have been collected. After each epoch the ANN is evaluated with a test set. The value of the Mean Square Error (MSE) obtained is used to perform Early Stopping and to retain the best models.

The conditions for overtraining and the maximum number of epochs are then verified. If they are true, the flag, which indicates that the threshold of quality has been reached, will also be verified and if it is on, the training of the inverse model starts, otherwise the models will be reset since new models need to be prepared.

After testing the conditions for overtraining and the maximum number of epochs, if they are both false, the predefined threshold of quality will also be tested and if it has been reached the variable Flag will be set to on. In either case the remaining time of the sampling period is tested to decide if a new epoch is to be performed or if a new sample is to be collected and training is to be performed with this new sample included in the sliding window.

The procedure to produce the inverse model is very similar and a similar block diagram could be used to represent it. The on-line training goes on switching from direct to inverse model each time a new model is produced. The main difference between the procedure for direct and inverse model lies in the evaluation step. While the direct model is evaluated with a simple test set, the inverse model is evaluated with a control simulation corresponding to the hybrid Direct/Specialized approach for generating inverse models (Dias et al., 2004).

As the measurements have noise the signals were filtered from high frequency noise.

During the on-line training the NNSYSID (Nørgaard, 1996b) and NNCTRL (Nørgaard, 1996a) toolboxes for MATLAB were used.

4. THE TEST SYSTEM

The test system chosen is a reduced scale prototype kiln for the ceramic industry, which is non-linear and will be working under measurement noise because of the type B thermocouple used.

The system is composed of a kiln, electronics for signal conditioning, power electronics module and a Data Logger from Hewlett Packard HP34970A to interface with a Personal Computer (PC) connected as can be seen in figure 3.

Through the Data Logger bi-directional real-time information is passed: control signal supplied by the controller and temperature data for the controller. The temperature data is obtained using a thermocouple. The power module receives a signal from the Data Logger, with the resolution of 12 bits (0 to 4.095V imposed by Data Logger), which comes from the controller implemented in the Personal Computer, and converts this signal in a Pulse Width Modulation (PWM) signal of 220V applied to the heating element.

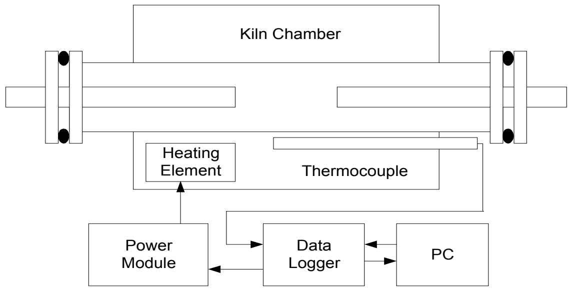

Figure 3 – Block diagram of the system.

The signal conversion is implemented using a sawtooth wave generated by a set of three modules: zero-crossing detector, binary 8 bit counter and D/A converter. The sawtooth signal is then compared with the input signal generating a PWM type signal.

The PWM signal is applied to a power amplifier stage that produces the output signal. The signal used to heat the kiln produced this way is not continuous, but since the kiln has integrator behaviour this does not affect the functioning.

The block diagram of the power module can be seen in figure 4.

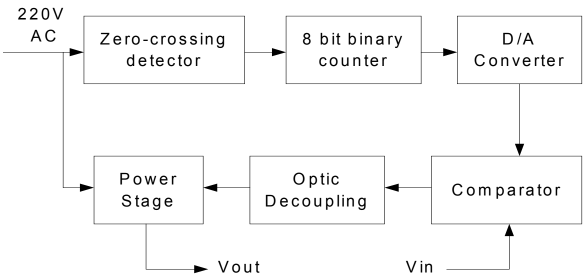

Figure 4 – Block diagram of the power module.

The Data Logger is used as the interface between the PC and the rest of the system. Since the Data Logger can be programmed using a protocol called Standard Commands for Programmable Instruments (SCPI), a set of functions have been developed to provide MATLAB with the capability to communicate through the RS-232C port to the Data Logger.

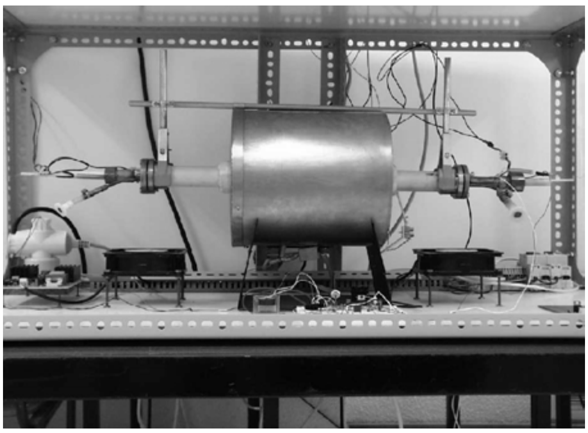

Figure 5 – Picture of the kiln and electronics.

Using the HP34902A (16 analog inputs) and HP34907A (digital inputs and outputs and two Digital to Analog Converters) modules together with the developed functions, it is possible to read and write values, analog or digital, from MATLAB. A picture of the system can be seen in figure 5. The kiln is in the centre and at the lower half are the prototypes of the electronic modules.

5. THE RESULTS OBTAINED

The static test set used for the present work can be seen in figure 6. This signal was selected because it covers the operating range quite well.

The test sequence is composed of 750 points although in the MSE calculation the first 50 points are not used since in the case of simulation of control. These initial values have no meaning because the simulation always starts from zero while the reference starts with the first value used.

The sliding window used for training is composed of 1250 samples, but the training started after 700 samples were collected.

Both direct and inverse models were one hidden layer models with 6 neurons on the hidden layer and one linear output neuron. The models have as inputs the past two samples of both the output of the system and the control signal.

The sampling period used was 30 seconds which allowed performing several epochs of training between each control iteration.

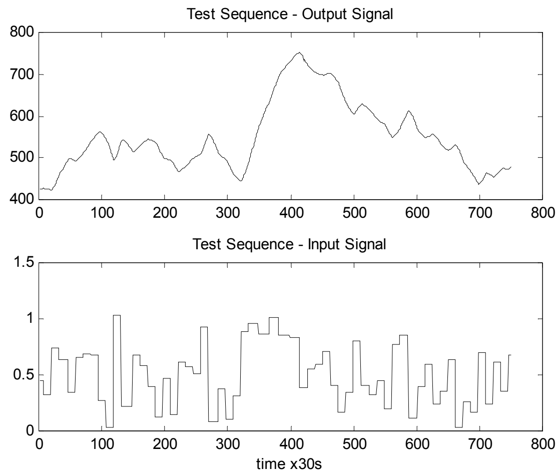

Figure 6 – Static Test Set used.

During the initial phase of collecting data a PID was used in order to keep the system operating within the range of interest. The PID parameters are Kp=25, Ki=0.004 and Td=25. After this initial phase the PID is replaced by a simple Direct Inverse Control (DIC), using the inverse model trained on-line. Figure 7 shows the logical diagram for DIC and the control results can be seen in figure 8.

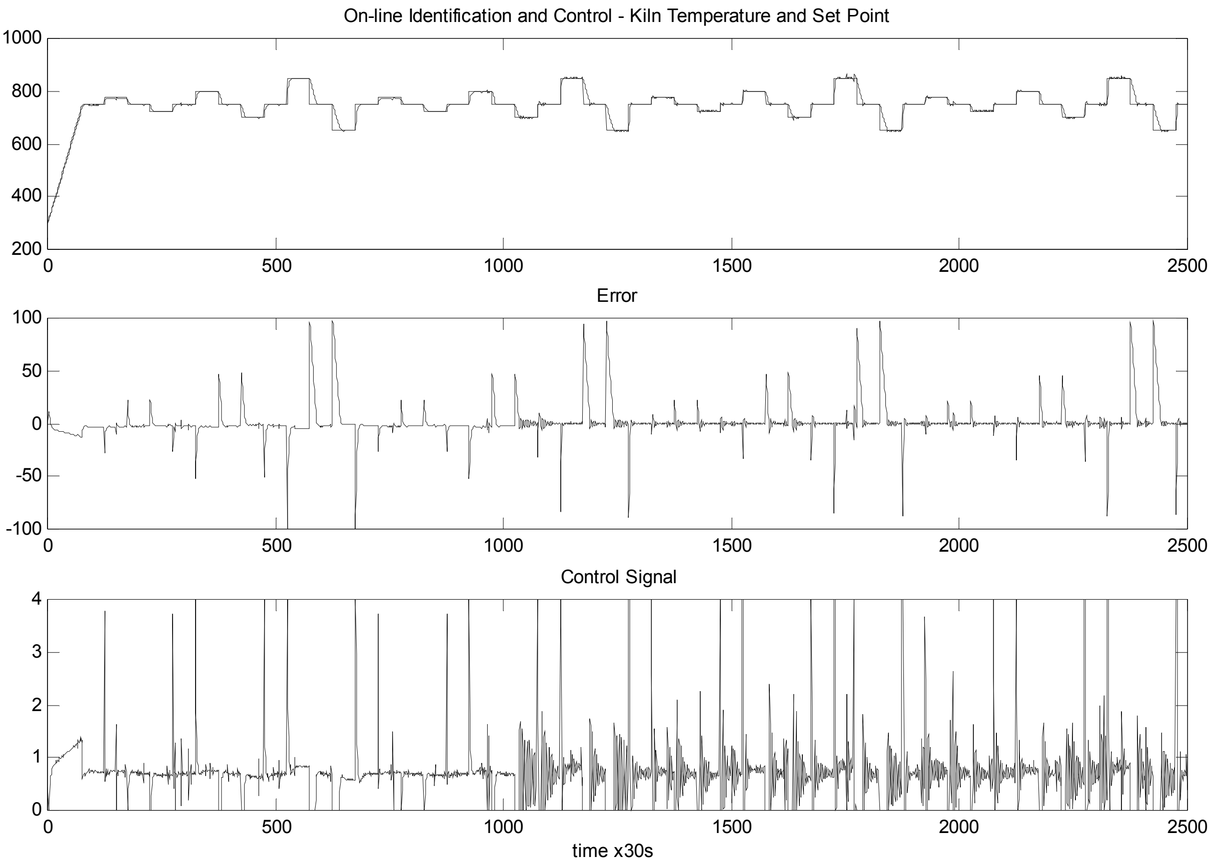

Figure 8 – On-line identification and control. The first part of the control is performed by a PID and the second by DIC.

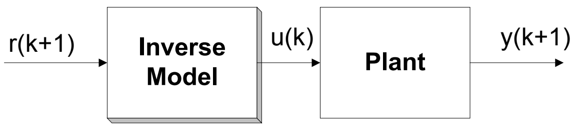

Figure 7 – Block diagram for Direct Inverse Control.

The first inverse model is ready at sample 1031 that is only 331 samples after the training has started. A first pair of models is ready to be used for control even though the Matlab code was running on a personal computer with a Celeron processor at 466MHz using 64Mbytes of memory.
Figure 9 shows the detail of the control signals scaled where the differences from the control of the PID and DIC types can be seen.
The PID is just used to maintain the system in the operating range while data is being collected and is disconnected as soon as the ANN models are ready.

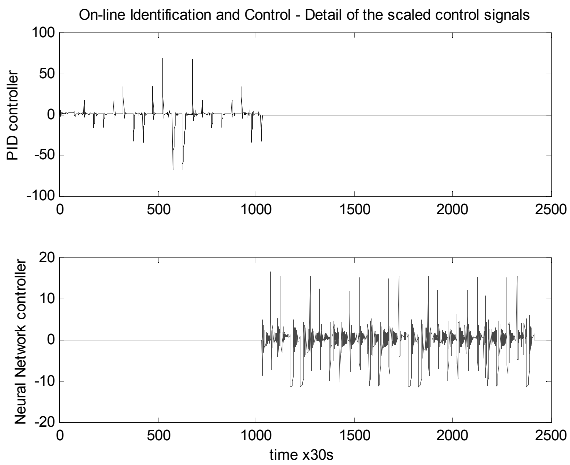

Figure 9 – Detail of the control signals. The upper half shows the control signal from the PID and the lower half is the one from the ANN.

6. CONCLUSIONS

This paper presents work that is still under development. In this initial stage on-line identification and control are performed using Early Stopping with a static test set.
The problems pointed out in section 3 to perform Early Stopping under a changing sliding window for the training set were not critical and a good choice of the parameters for identification of the overtraining situation and for the maximum number of iterations for each attempt to create a model were sufficient to obtain reasonable models to perform DIC.
As it is easily noticeable from figure 8, the DIC originates a typical oscillation or ringing in the control signal. Nevertheless, the quality of control is increasing as the number of samples advances and better models are produced.
The sliding window approach with Early Stopping solves the problems for using the Levenberg-Marquardt algorithm on-line due to the difficulty of creating a true iterative version, which includes the computation of the trust region.
As shown here, even for a noisy system, for which overtraining is a real problem it is possible to create models on-line of acceptable quality.

7. REFERENCES

Levenberg, K., “A method for the solution of certain problems in least squares”. Quart. Appl. Math., 2:164—168, 1944.

Marquardt, D., “An algorithm for least-squares estimation of nonlinear parameters”, SIAM J. Appl. Math., 11:431—441, 1963.

Nørgaard, M., O. Ravn, N. K. Poulsen, and L. K. Hansen, “Neural Networks for Modelling and Control of Dynamic Systems. Springer, 2000.

Ferreira, P., E. Faria, and A. Ruano, “Neural network models in greenhouse air temperature prediction”, Neurocomputing, 43, no. 1-4:51-75, 2002.

Ngia, Lester S. H., “System Modeling Using Basis Functions and Application to Echo Cancelation”, PhD thesis, Department of Signals and Systems School of Electrical and Computer Engineering, Chalmers University of Technology, 2000.

Morgan, N. and H. Bourlard, “Generalization and parameter estimation in feedforward nets: Some experiments”, Advances in Neural Information Processing Systems, Ed. D.Touretzsky, Morgan Kaufmann, pages 630—637, 1990.

Sjöberg, J., “Non-Linear System Identification with Neural Networks”, PhD thesis, Dept. of Electrical Engineering, Linköping University, Suécia, 1995.

Nørgaard, M. (1996a), “System Identification and Control with Neural Networks”, PhD Thesis, Department of Automation, Technical University of Denmark.

Nørgaard, M. (1996b), “Neural Network System Identification Toolbox for MATLAB”, Technical Report.

Nørgaard, M. (1996c), “Neural Network Control Toolbox for MATLAB”, Technical Report.

Dias, F. M., A. Antunes and A. Mota, “A new hybrid direct/specialized approach for generating inverse neural models”, 6th WSEAS Int. Conf. on Algorithms, Cientific Computing, Modelling and Computation (ASCOMS’04), Cancun, 2004.