International conference on Research Reactors: Safe Management and Effective Utilization, 5 – 9 November 2007 – Sidney – Australia

ENEA Italian Agency for Energy, New Technologies and the Environment – Italy

Neural Networks application to CRDM and thermo-hydraulic data validation

Authors: M. Sepielli[1], M. Palomba[2], A. Ratto[3], R. Rosa[2], M. Bernabucci[3], [1]ENEA – FPN [2]ENEA – FPN-FISON [3]ENEA guests

Reference: Ing. Massimo Sepielli – ENEA – FPN – Via Anguillarese, 301 – 00123 Rome - Italy

Summary

The work is carried out on ENEA RC-1, a Triga Mark II 1 MW - demineralized water / natural convection cooled - research reactor, located at ENEA Casaccia Centre. Activity is aimed at intelligent processing of the data obtained by reactor measurements through soft-computing models based on neural networks (NN). A first application is made for CRDM rod position validation and a second one is devoted to the fuel temperature prediction. Both use the NN-based algorithms working on real data coming from sensors. The bias between real and calculated values is used to train the NNs for improving the performance in the further experiments.

1. Introduction

Continuous remote monitoring is more and more needed as the new generation reactors tend to drammatically decrease the outage time for maintenance and refuelling operations.
Plant sensor on-line monitoring, data validation through soft-computing process and plant condition monitoring techniques would help identify plant sensors drift or malfunction and operator actions in addressing nuclear reactor control. On-line recalibration can often avoid intervene with manual calibration or physical replacement of the drifting component.

Further, neuro-fuzzy logic allows to link field measurement and physical system behaviour, so interpreting the data coming from the reactor instrumentation and revealing eventual mismatch due to miscalibration or sensoring faults.

The activity described in the paper is aimed at validating data obtained by reactor measurements through soft-computing models based on neural networks (NN).
Application is made on the CRDM rod position and the fuel temperature control which are both measured and re-calculated by NN-based algorithms and compared with the real data coming from sensors. The bias is used to train the NNs for improving the performance in the next experiments.

2. Description of the reactor

The reactor is a typical TRIGA light-water (research) reactor with an annular graphite reflector cooled by natural convection, with a power of 1MW.
The reactor core is placed at the bottom of the 6.25-m-high open tank with 2-m diameter. The core has a cylindrical configuration.
In total there are 91 locations in the core, which can be filled either by fuel elements or other components like control rods, a neutron source, irradiation channels, etc. The core lattice has an annular but not periodic structure.
The core temperature is measured by 20 thermocouple situated above and under the core (see Fig.1), while the fuel temperature is measured in two fuel elements instrumented with thermocouples.

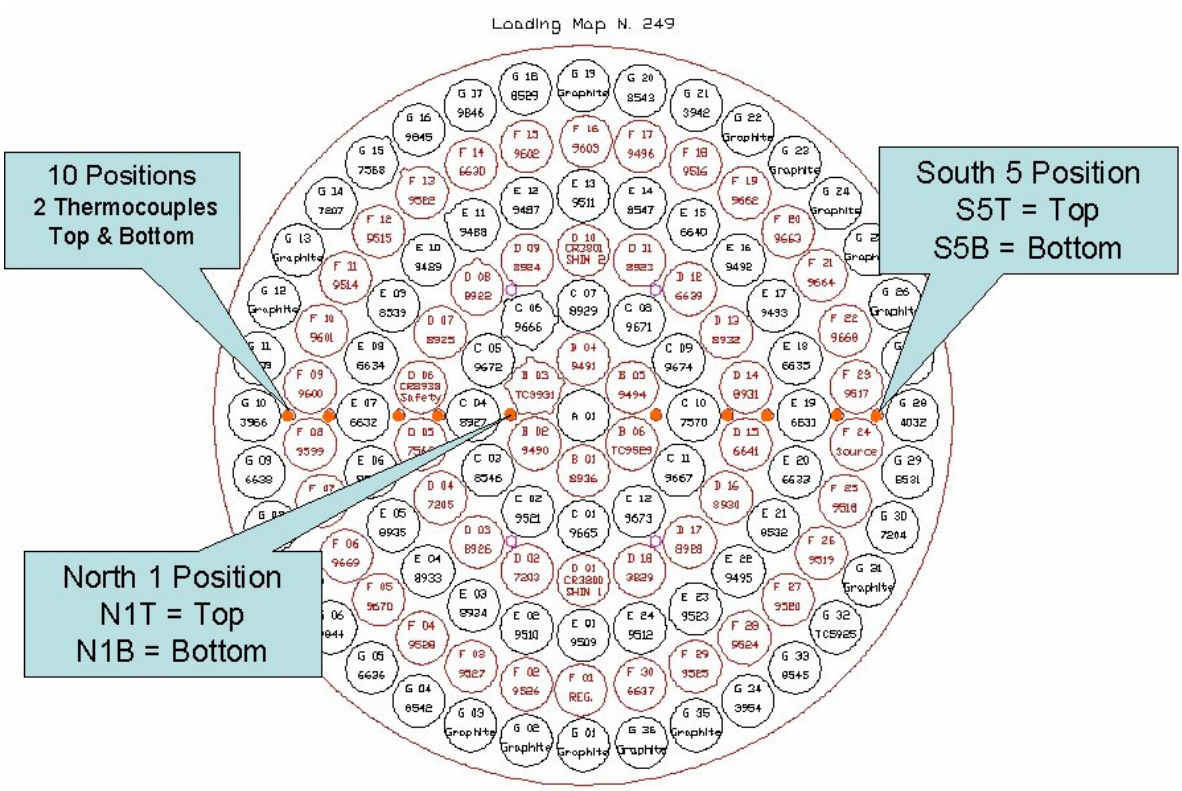


Fig.1 – TRIGA Core: thermocouples position

3. Neural Network application to validation of control rod position

In this first application the position of the control rod is being validated. In order to, a neural network elaborates the data measured on the thermo-fluid-dynamic system of the reactor, in parallel with the control system.
The net is feed-forward 3-layers network trained through second-order algorithms from Levenberg-Marquart.

It consists of two nets, see Fig.2, working in parallel: a first one is used for data validation and initially trained through the use of reactor data set; a second one is used for training and updates its parameters every time a given error occurs due to the difference between the exact value generated by the first net and the value measured from the instruments and validated by the operator.
The new training parameters will then be transferred to the first net.

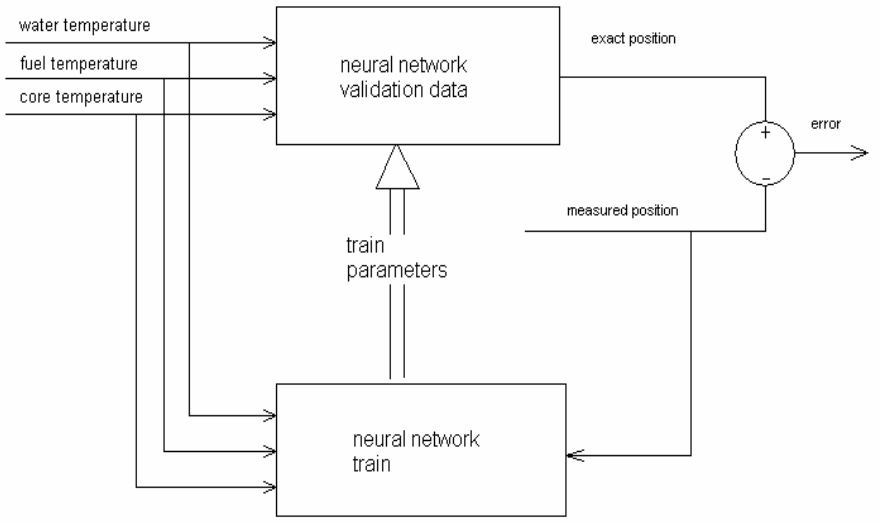


Fig.2 Validation network

The geometry ot the two nets is common and consists of a 35 neurons input layer, a 22 neurons hidden layer and a 2 neurons output layer, see Fig.3.

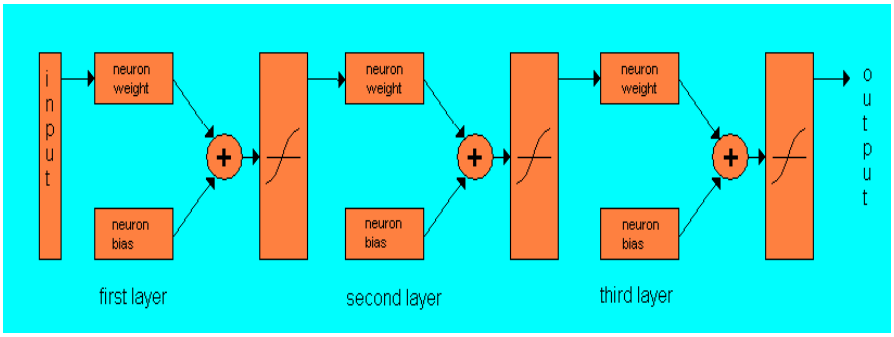


Fig.3 Neural network structure

3.1 Levenberg-Marquardt Algorithm

The Levenberg-Marquardt (LM) algorithm is an iterative technique that locates the minimum of a multivariate function that is expressed as the sum of squares of non-linear real-valued functions.

LM can be thought of as a combination of steepest descent and the Gauss-Newton method. When the current solution is far from the correct one, the algorithm behaves like a steepest descent method: slow, but guaranteed to converge. When the current solution is close to the correct solution, it becomes a Gauss-Newton method.

Let f be an assumed functional relation which maps a *parameter vector* $\mathbf{p} \in R^m$ to an estimated *measurement vector* $\underline{\mathbf{x}} = f(\mathbf{p})$, $\underline{x} \in R^n$. An initial parameter estimate $\mathbf{p}_0$ and a measured vector $\underline{x}$ are provided and it is desired to find the vector $\mathbf{p}^+$ that best satisfies the functional relation f, i.e. minimizes the squared distance $\varepsilon^T\varepsilon$ with $\varepsilon = \mathbf{x} - \underline{\mathbf{x}}$. The basis of the LM algorithm is a linear approximation to f in the neighborhood of $\mathbf{p}$. For a small $\|\delta_p\|$, a Taylor series expansion leads to the approximation

$$f(\mathbf{p} + \delta_p) \approx f(\mathbf{p}) + \mathbf{J}\delta_p \tag{1}$$

where $\mathbf{J}$ is the Jacobian matrix.

Like all non-linear optimization methods, LM is iterative: Initiated at the starting point $\mathbf{p}_0$, the method produces a series of vectors $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, … that converge towards a local minimizer for $\mathbf{p}^+$. Hence, at each step, it is required to find the δ_p that minimizes the quantity

$$\|\mathbf{x} - f(\mathbf{p} + \delta_p)\| \approx \|x - f(\mathbf{p}) + \mathbf{J}\delta_p\| = \|\varepsilon - \mathbf{J}\delta_p\|.$$

The sought δ_p is thus the solution to a linear least-squares problem: the minimum is attained when $\mathbf{J}\delta_p - \varepsilon$ is orthogonal to the column space of $\mathbf{J}$. This leads to

$$\mathbf{J}^T(\mathbf{J}\delta_p - \varepsilon) = 0$$

which yields δ_p as the solution of the so-called *normal equations*:

$$\mathbf{J}^T\mathbf{J}\delta_p = \mathbf{J}^T\varepsilon \tag{2}$$

The matrix $\mathbf{J}^T\mathbf{J}$ in the left hand side of Eq. (2) is the approximate Hessian, i.e. an approximation to the matrix of second order derivatives. The LM method actually solves a slight variation of Eq. (2), known as the *augmented normal equations*

$$\mathbf{N}\delta_p = \mathbf{J}^T\varepsilon \tag{3}$$

where the off-diagonal elements of $\mathbf{N}$ are identical to the corresponding elements of $\mathbf{J}^T\mathbf{J}$ and the diagonal elements are given by

$$\mathbf{N}_{ii} = \mu + [\mathbf{J}^T\mathbf{J}]_{ii}$$

for some $\mu > 0$. The strategy of altering the diagonal elements of $\mathbf{J}^T\mathbf{J}$ is called *damping* and μ is referred to as the *damping term*. If the updated parameter vector $\mathbf{p} + \delta_p$ with δ_p computed from Eq. (3) leads to a reduction in the error ε , the update is accepted and the process repeats with a decreased damping term. Otherwise, the damping term is increased, the augmented normal equations are solved again and the process iterates until a value of δ_p

that decreases error is found. The process of repeatedly solving Eq. (3) for different values of the damping term until an acceptable update to the parameter vector is found corresponds to one iteration of the LM algorithm.

In LM, the damping term is adjusted at each iteration to assure a reduction in the error ε . If the damping is set to a large value, matrix **N** in Eq. (3) is nearly diagonal and the LM update step δ_p is near the steepest descent direction. Moreover, the magnitude of δ_p is reduced in this case. Damping also handles situations where the Jacobian is rank deficient and $\mathbf{J}^T\mathbf{J}$ is therefore singular [3]. In this way, LM can defensively navigate a region of the parameter space in which the model is highly nonlinear. If the damping is small, the LM step approximates the exact quadratic step appropriate for a fully linear problem. LM is adaptive because it controls its own damping: it raises the damping if a step fails to reduce ε ; otherwise it reduces the damping. In this way LM is capable to alternate between a slow descent approach when being far from the minimum and a fast convergence when being at the minimum's neighborhood [3].

The LM algorithm terminates when at least one of the following conditions is met:

- The magnitude of the gradient of $\varepsilon^T\varepsilon$, i.e. $\mathbf{J}^T$ in the right hand side of Eq. (2), drops below a threshold ε_1;
- The relative change in the magnitude of δ_p drops below a threshold ε_2
- The error $\varepsilon^T\varepsilon$ drops below a threshold ε_3
- A maximum number of iterations k_{max} is completed

3.2. NN Training

3.2.1 Training data set

Thermo-hydraulic values of the system are varied with the position of the control rod.

The aim of this first application is the validation of control rod position. In order to do this, core temperature, fuel temperature and water temperature are selected as meaningful variables.

The training data set for neural network training is retrieved by the reactor data capture software which records all values every 4 seconds. It is composed by:

Input data: core temperature (20 thermocouple); fuel temperature; water temperature.

Output data: rod position (2 channels).

Figures 4 and 5 display, the core temperature trends from thermocouples N3T N3B, S2T S2B, as a significant example of the whole 20 thermocouples set.

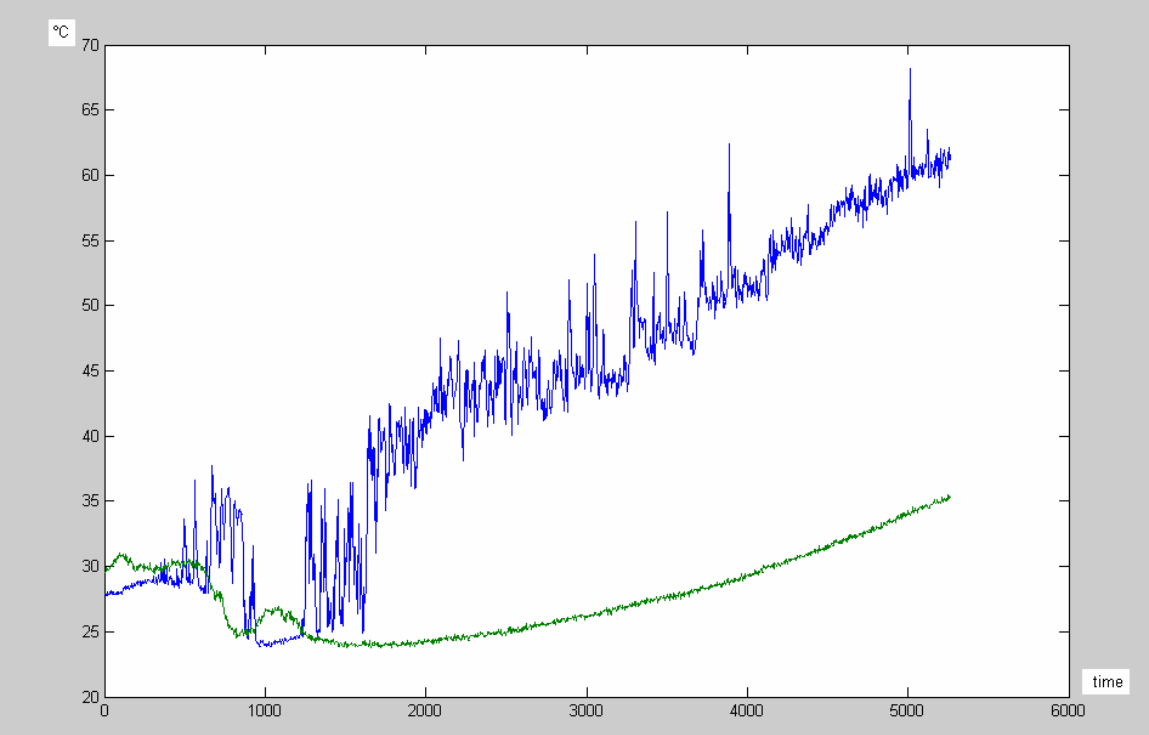


Fig.4 Core temperature - N3 Top (green) Bottom (bleu)

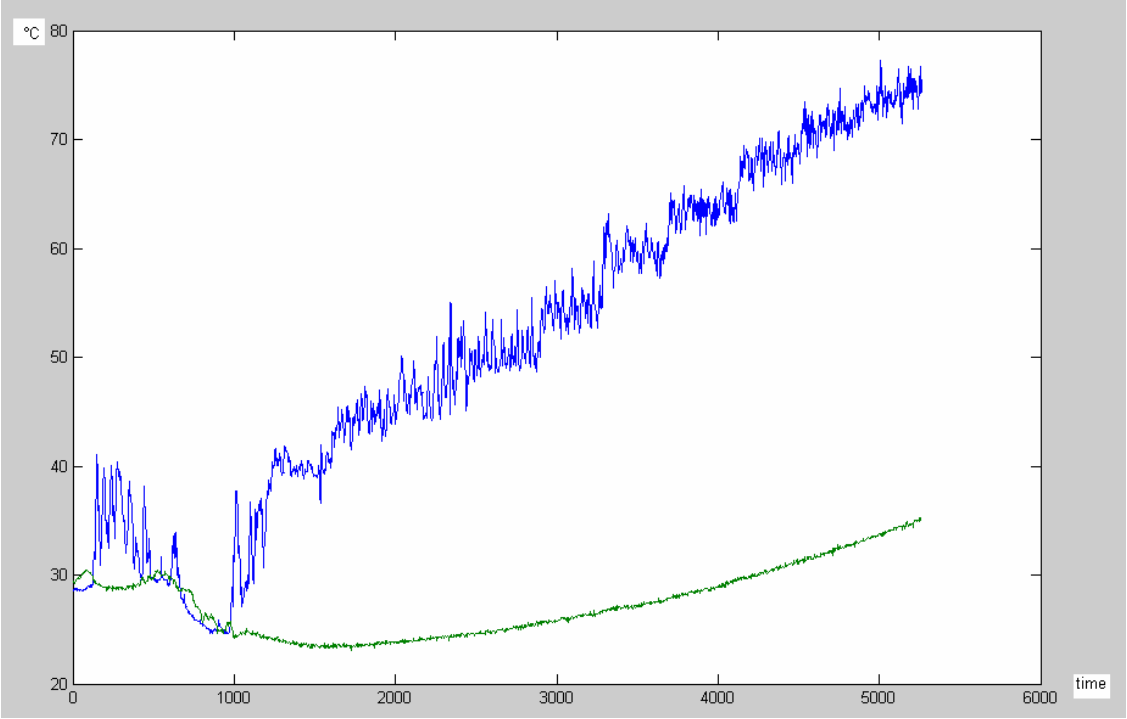


Fig.5 Core temperature S2 Top (green) Bottom (bleu)

In Figure 6 the trend of fuel and water temperatures is shown, according to the control rod withdrawal and the power raise.

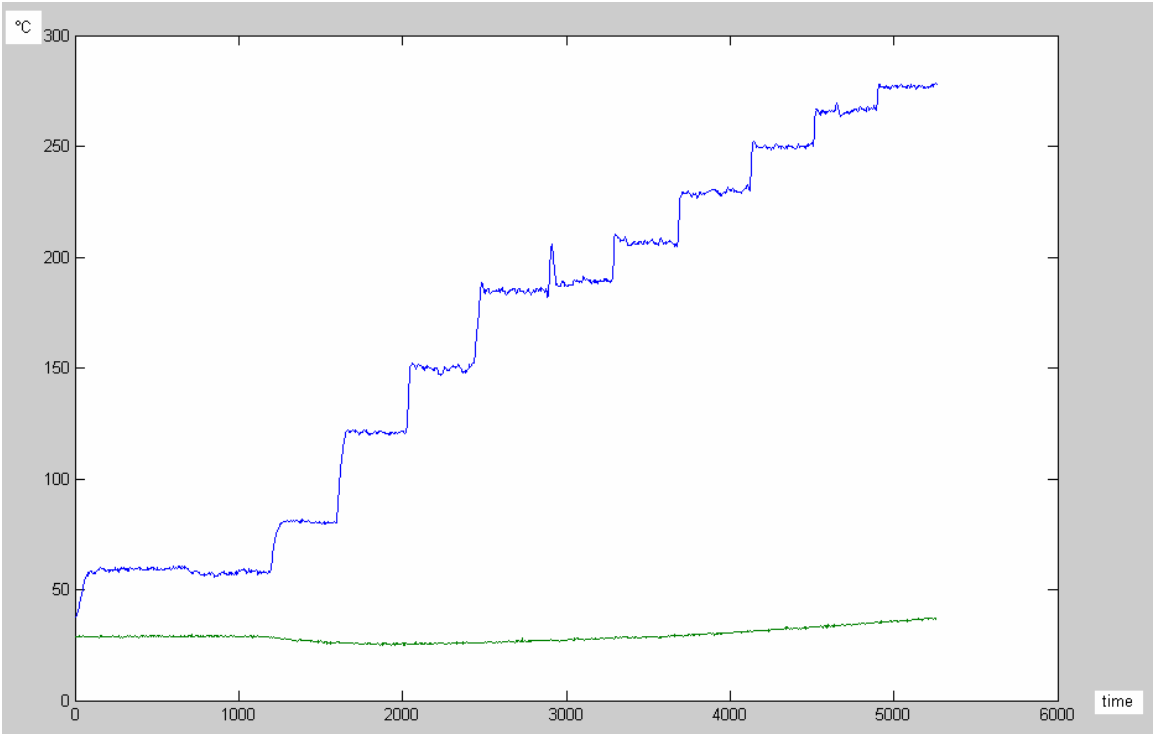


Fig.6 Fuel Temperature (bleu) Water Temperature (green)

In Figure 7 the channels SH1 and SH2 rod position is shown.

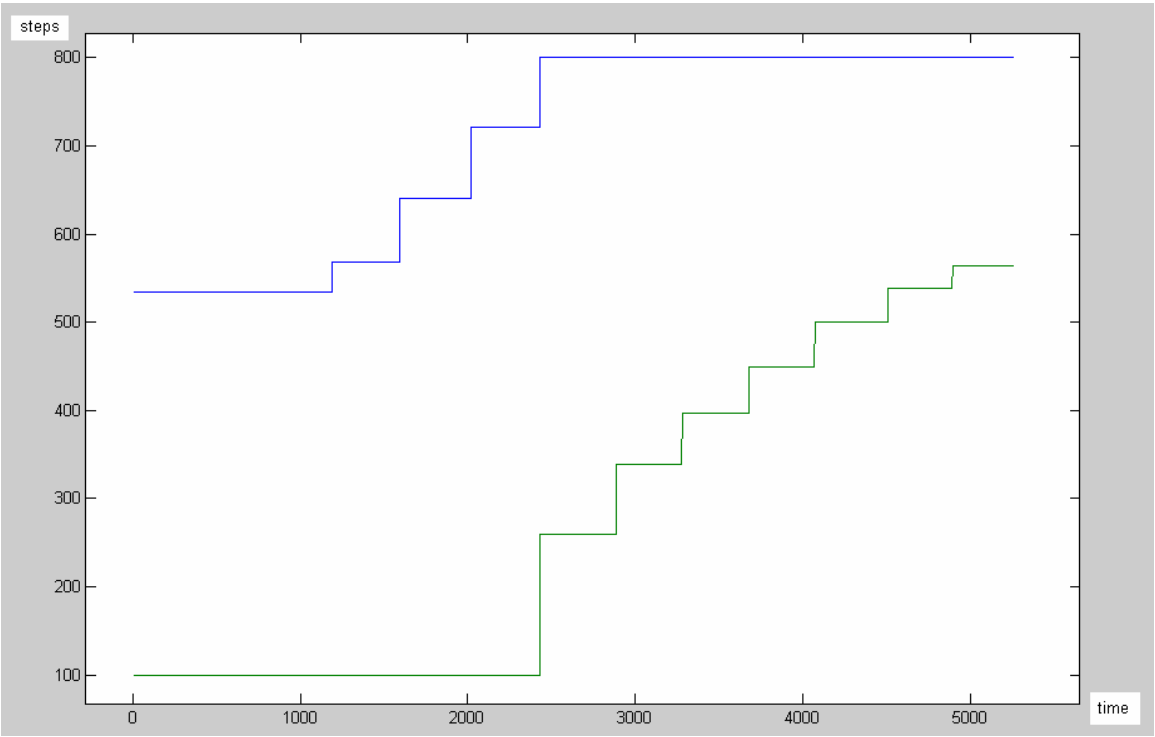


Fig.7 Rod position SH1 (green) SH2 (bleu)

3.3 Training

In order to implement the previously described L.M algorithm., ***TRAINLM*** function of Matlab is used.

The chosen parameters for training the net have been:

- The magnitude of the gradient: 10^{-10} ;
- The relative change in the magnitude: @@@@
- The error: 0
- A maximum number of iterations: 140 epochs

After the training process, the value of the achieved error has been 9.8e-8.

It has been noticed, either, that increasing the number of the epochs, the output data remain almost unchanged.

The error's trend is shown, below, in Fig.8.

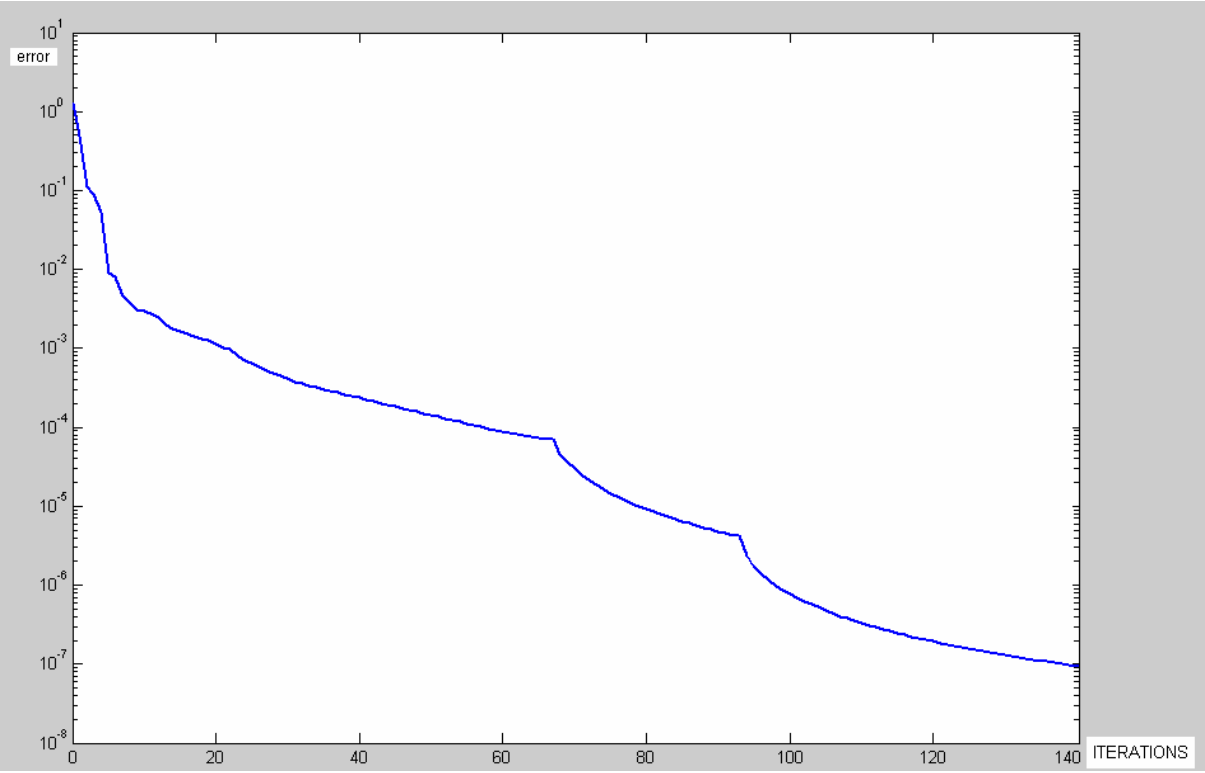


Fig.8 Error's trend

As the error resulted to be sufficiently low, the choice of used data demonstrated to be appropriate for neural network training.

The outcome of training is displayed in figg. 9 and 10, below.

In Fig.9 it's shown that the experimental data match with neural network data.

In fig. 10, it's shown how the rod position error remains low,ranging between 10^{-2} and 10^{-3} steps.

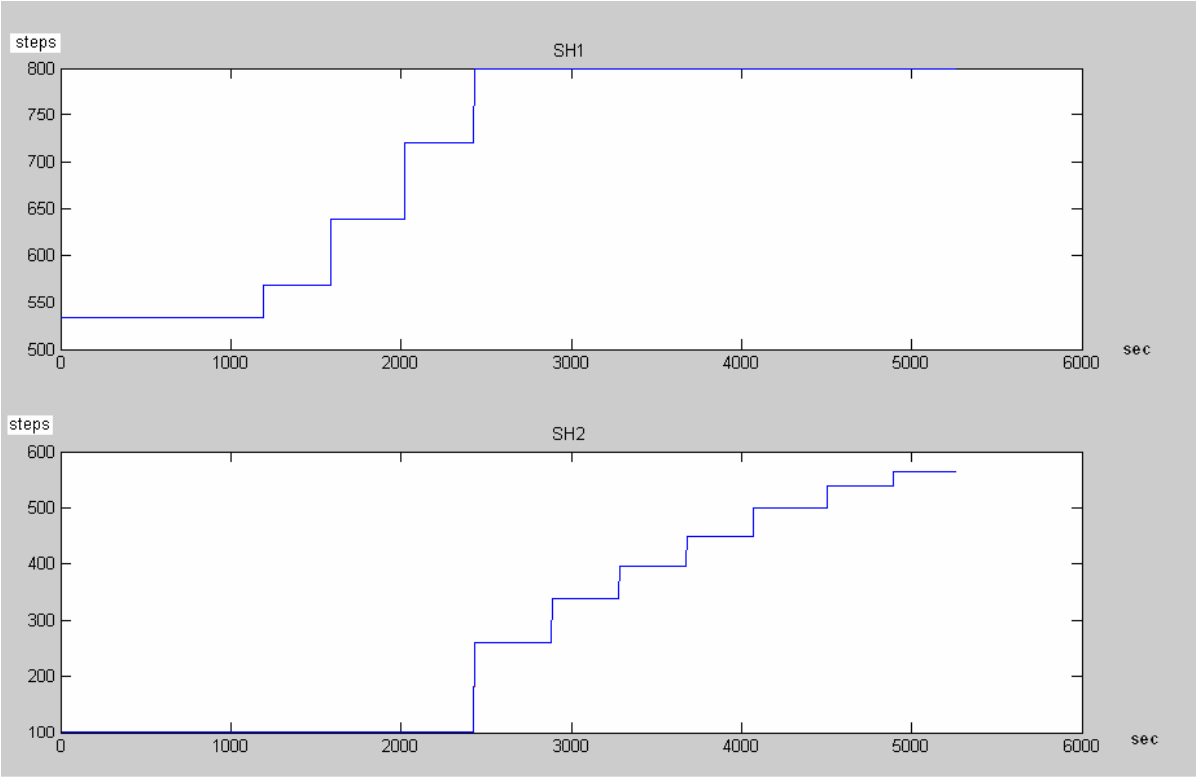


Fig.9 Rod position channels

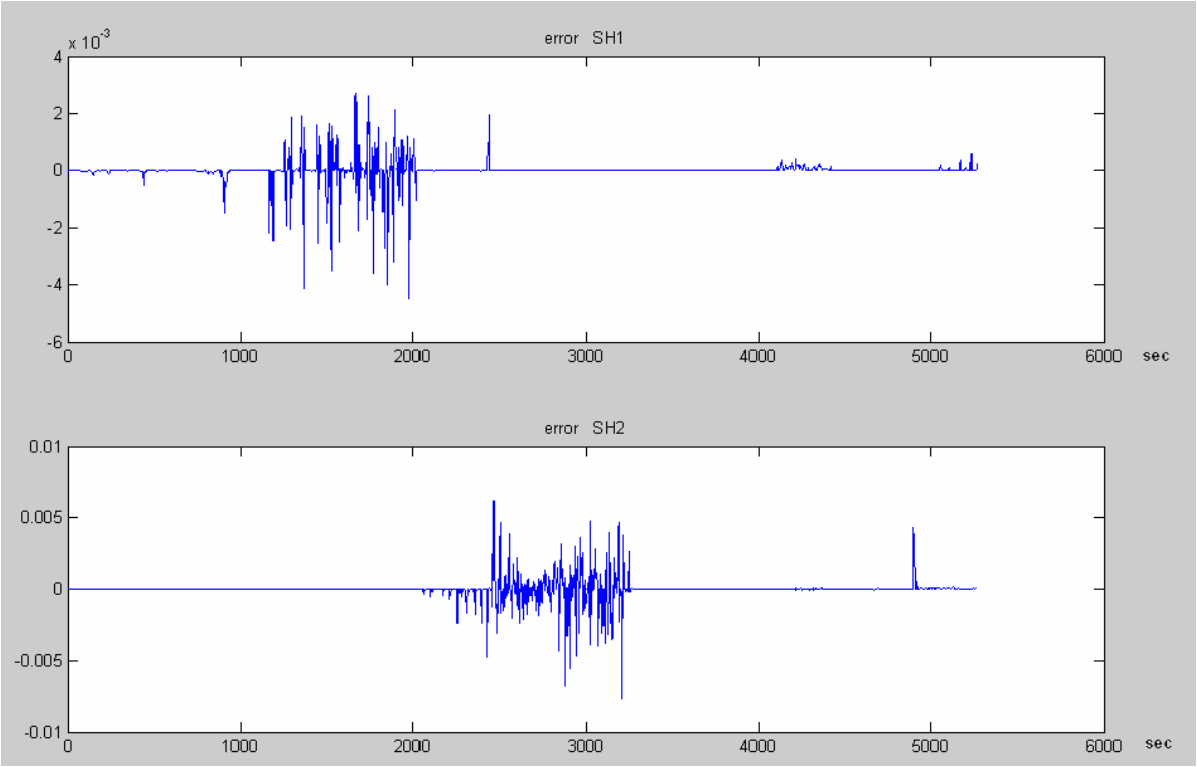


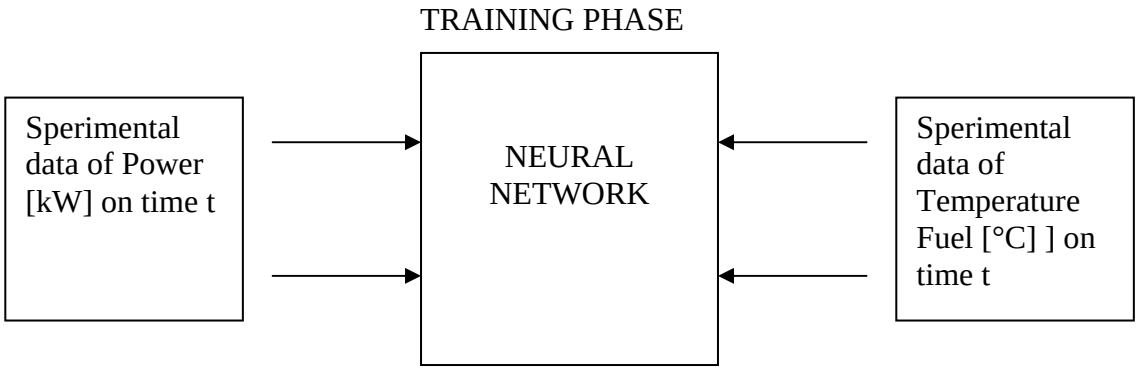
Fig.10 Rod position error

4. Neural Network application to fuel temperature condition

In this second application we want predict the value of the fuel temperature for assigned value of power. To do this we use a neural network trained with the measured power and fuel temperature.
The used net is feed-forward network trained using supervised learning algorithm (back-propagation with steepest descent) implemented in the code-program.

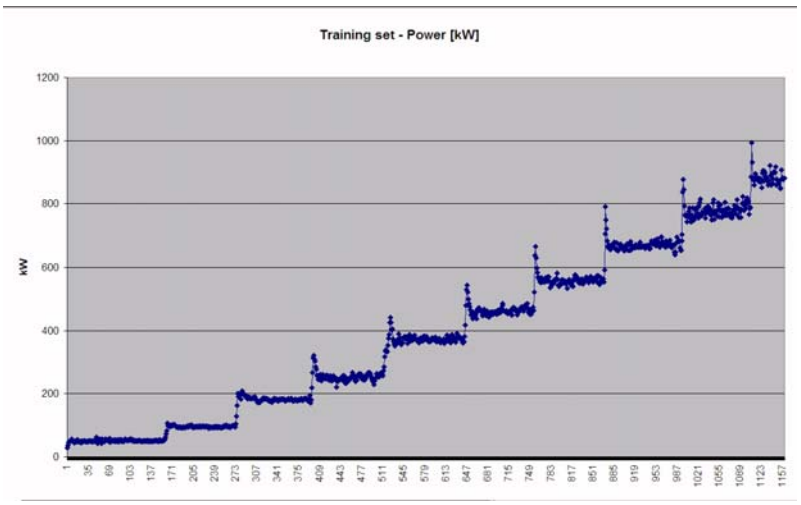
Training

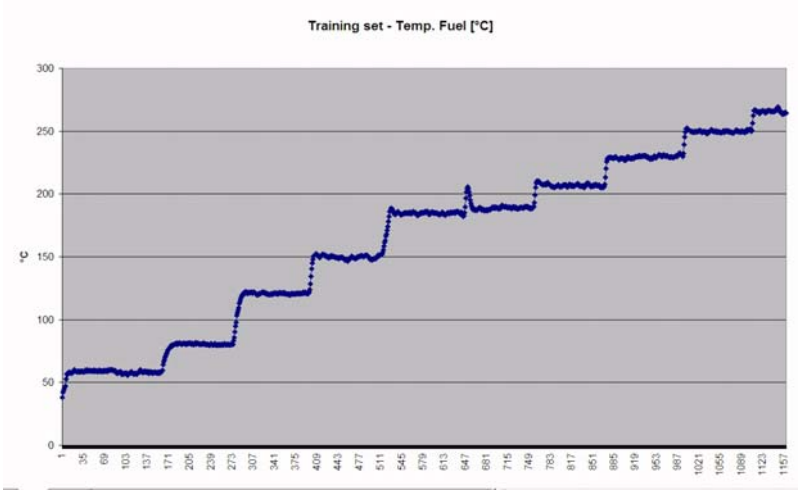
Supervised learning algorithms adjust synaptic weights using input–output data to match the input–output characteristics of a network to desired characteristics.



The training set is formed of 1161 couple (Power, Temp.fuel). One complete presentation of the entire training set during the learning process is called an "epoch". In this case on presented 20 epoch. The structure of neural network is two internal layer and tree neuron for level.

The training set is displayed below

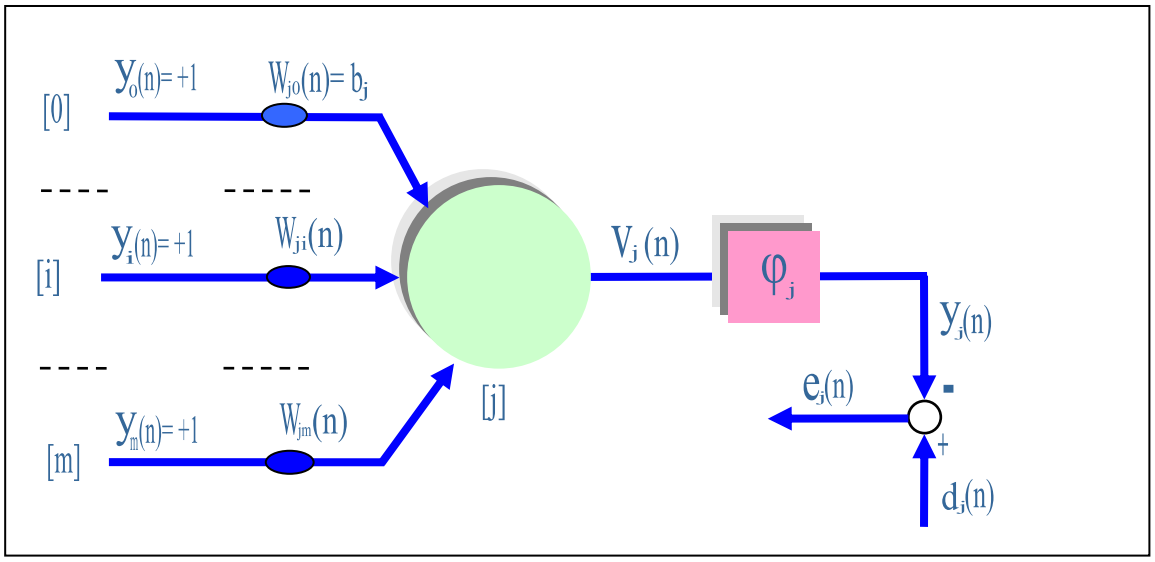




Learning Algorithm

The error signal at the output of neuron j at iteration n (i.e., presentation of the nth training example) is defined by:

$$e_j(n)=d_j(n)-y_j(n) \qquad (1)$$



We define the instantaneous value of the total error energy over all neurons in the output layer is:

$$E(n)=\frac{1}{2}\sum_{j\in V} e_j^2(n) \qquad (2)$$

Let N denote the total number of patterns (examples) contained in the training set. The *average squared error energy* is obtained by summing E(n) over all *n* and then normalizing with respect to the set size *N*, as shown by

$$E_{av}=\frac{1}{N}\sum_{n=1}^{N} E(n)$$

E_{av} is a index of of the level of learning.
Since NN outputs are changed when synaptic weights are modified, the E(n) must be a function of the synaptic weights **w**
Since E(n) is a function of **w**, the searching direction of the smaller error point is obtained by calculating a partial differential. The searching direction **g**=∂E(n,**w**)/∂**w** and modification of weights is given as

$$\Delta \mathbf{w} = \eta \mathbf{g}$$

Where η is learning-rate parameter.

In the code program the back-propagation algorithm is implemented

Defined the *local gradient*:

$$\delta_j(n) = -\frac{\partial E(n)}{\partial v_j(n)} = e_j(n) \cdot \varphi'_j[v_j(n)]$$

for modified weights:

$$\Delta w_{ji} = \eta \cdot \delta_j(n) \cdot y_i(n)$$

For the characteristic function φ() of neutron unit on used hyperbolic tangent

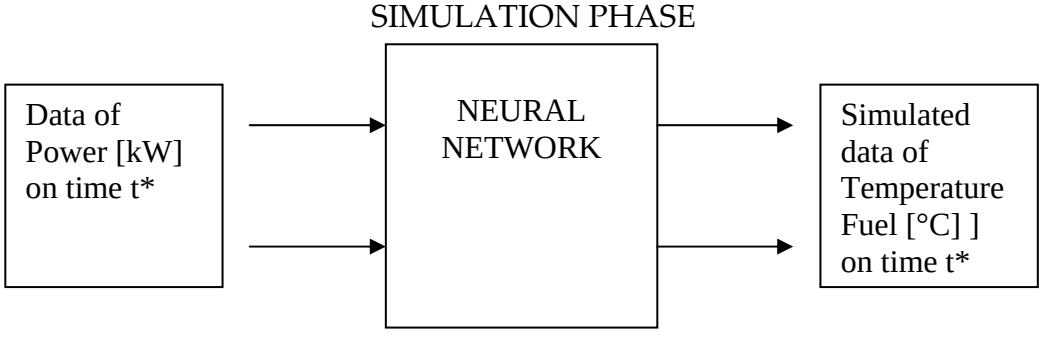
$$\varphi_j(v_j) = a \cdot tgh(b \cdot v_j) \qquad a,b > 0$$

$$\varphi'_j(v_j) = a \cdot b \cdot sech^2(b \cdot v_j) = \frac{b}{a}[a^2 - y_j^2(n)]$$

where a=1.7159; b=2/3 [*]

<u>Prediction</u>

In the simulation phase the temperature values have been demanded from the net correspondenting to the power values which were introduced.
The predicted set is external to the training data range. The error percentage between the simulated data and comparison experimental data depends on the numerousness of the training set . When widening the training set a correndonding reduction of the error excursion is observed.



The outcome of simulation is displayed below

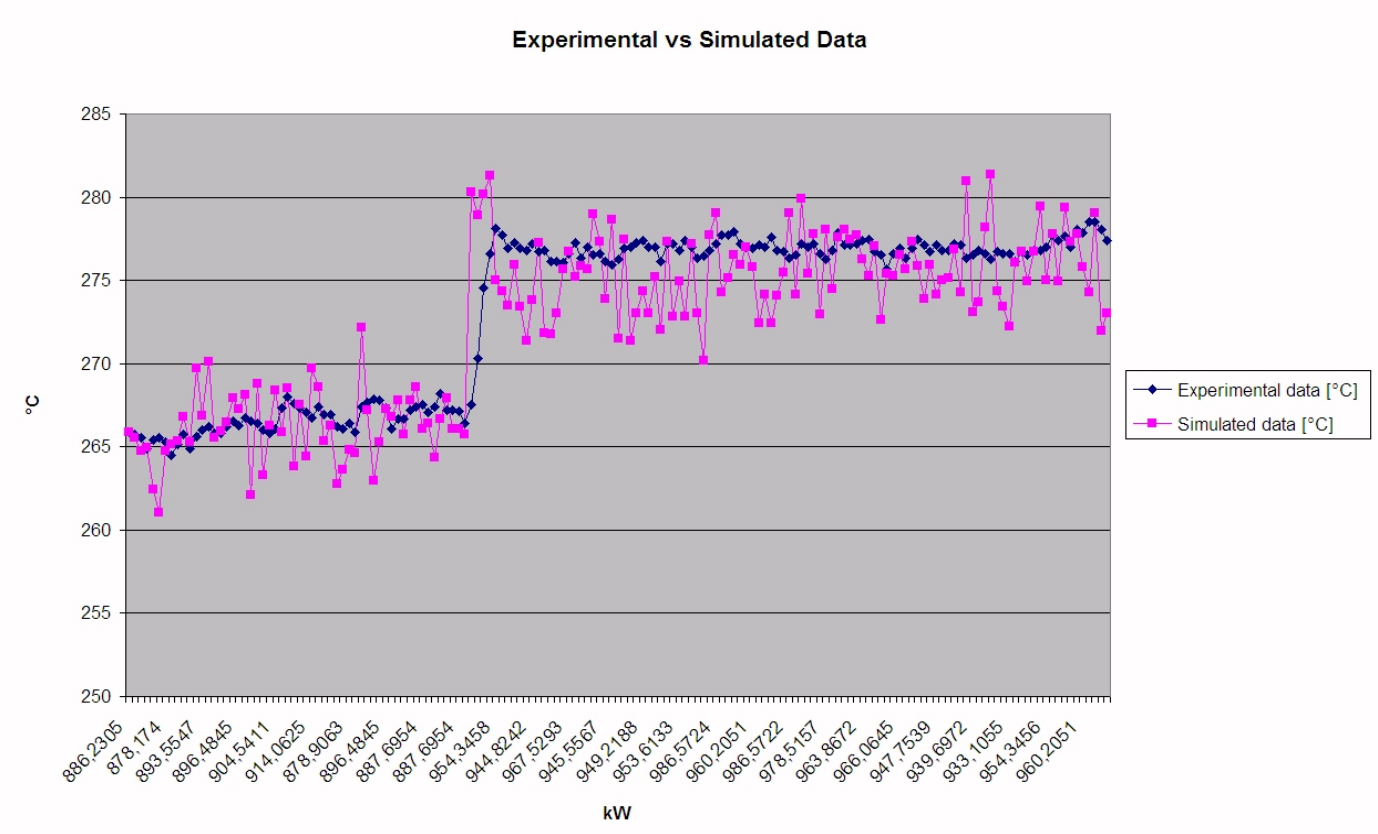


Fig.5 Experimental data (blue) Simulated data (pink)

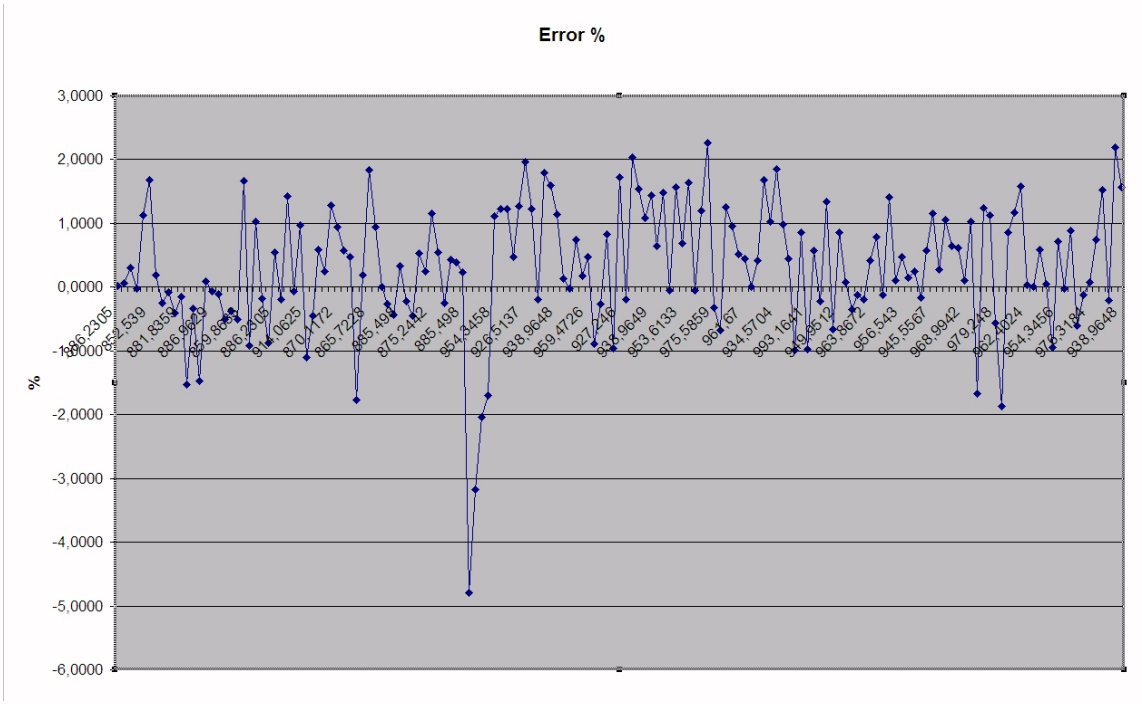


Fig.6 Percentage error

Remarks from second application

Experience aimed at appraising the feasibility of a system based on neural network for the forecast of an experimental device behavior. The results have satisfied the expectations. Following step is concerned with the optimization of the net parameters and the learning algorithm.
The potential for the application for neural network technology in the process industries is vast. The ability of neural network to capture and model process dynamics and severe process non-linearities make them powerful tools in model based control and monitoring.

General conclusions

The outcome obtained in this two applications have been satisfactory as the error in steady state resulted less than the expected one and the training method quite effective. Testing will continue with increasing of data scanning rate to improve the answer during status transitions and investigate how to decrease oscillations during the steady-states.

Further activity will concern the application of the method to validation and thermohydraulic prediction in a III+ generation light water nuclear power plant featured with an integral pressurised primary system, where access to CRDM system is physically hampered and rod positioning can be accurately and safely controlled from exterior only.

Bibliography

[*] Simon Haykin – Neural Networks a comprehensive foundation. Second Edition. Prentice Hall

Introduction to Fuzzy Systems, Neural Networks, and Genetic Algorithms - Hideyuki TAKAGI -Kyushu Institute of Design

Baldi, Hornik, 1989 – Neural networks and principal component analysis: learning from examples without local minimum

Battiti 1992 – First and second order methods for learning: between steepest descent and Newton's method.

Narendra 1992 – Adaptive control of dynamical system using neural networks

Y. J.Wang, J. Tu, 1999 – Neural network control, Beijing, Machinery industry press

A. G. Parlos, S. Parthasarathy, A. F. Atiya – Neuro-predictive process control using on-line controller adaptation

T. W. Miller, R. S. Sutton, P. J. Werbos, 1990 – Neural networks for control. Cambridge, MA: MIT Press.

K. F. Muller, W. F.Schaefer 1993 – Self-tuning control technology for nuclear power plants: Final rep, EPRI TR-101 650

M. T. Hagan H. B Demuth – Neural Networks for control

P. Haley, D. Soloway, B. Gold, Real-time adaptive control using neural generalized predictive control

M.Sepielli , Application of Neuro-Fuzzy Logic for Early Detection and Diagnostic in Gas Plants and Combustion Chambers at ENEA

M. T. Hagan, and M. Menhaj, Training feedforward networks with the Marquardt algorithm.

K. Levenberg, A method for the solution of certain problems in least squares.

M. Lampton, Damping-Undamping Strategies for the Levenberg-Marquardt Nonlinear Least-Squares Method.

D.W. Marquardt. An Algorithm for the Least-Squares Estimation of Nonlinear Parameters