

# BUILDING I/O NEURAL SIMULATOR of NUCLEAR POWER REACTORS

A. ABO-SHOSHA , M. ASHOUR, and F. MOHAMED  
ATOMIC ENERGY AUTHORITY (EGYPT)  
P.O.Box 29, Nasr City, Cairo, Egypt. Fax ++202-2749298

## ABSTRACT

In this paper, a neural simulator of the nuclear power reactor (NPR) will be presented. The simulator has been built using the multi-layer neural network (MNN) to simulate the NPR input-output (I/O) behavior. The MNN has been trained using the error back propagation training algorithm (EBPTA). To speed up the network's training process and to improve the precision, parameters of the learning process have been optimized. The resultant simulator is reliable and it has a simple design. The simplicity and the robustness of neural simulators helps in constructing and developing of simulators for several dynamic linear or even non-linear systems.

## KEYWORDS

*NPR I/O modeling and simulation, artificial neural networks ANNs, System dynamics*

## I. INTRODUCTION

One of the most important applications of Artificial Neural Networks (ANNs) is the simulator and model building [1]. In this case, the analyst designs a neural simulator, which can be used instead of the original system. This can be done by learning the network to generate alike system response if it is driven by the same system inputs (I/O simulation). The network follows the system response with the minimum error as far as the learning rule allows. Using neural simulators is wise method for analysis and study of large scale and complicated systems such as the NPR. By using neural simulators, we can study the system dynamics in all its operation conditions without depending on the actual system directly. That is a safe and an easy way to study the different characteristics and behaviors of the system under study without causing any system troubles due to testing procedures. The main goals of the learning process is to minimize errors, and to speedup the learning process. This can be done by optimizing network parameters to have appropriate values and to get tight rules. In section (II) of this paper, the reactor modeling and dynamics are presented in. Section (III) illustrates the EBPTA learning rule. Section (IV) presents the results of the simulation process. Section (V) demonstrates the enhancement of the EBPTA learning rule. Finally, section (VI) presents the conclusion and the discussion.

## II. REACTOR MODEL

The NPR is a large scale and complicated system. Fig.(1) shows the configuration of a traditional NPR station. The observed state of this system  $y_{(n)}$  is the reactor output thermal power, while the control element, called reference input,  $r_{(n)}$  is the control rod speed. The auto regressive exogenous (ARX) scheme models the relation between the output and the input in addition to the Gaussian white noise  $\varepsilon$ . The NPR dynamics model is represented with a one delayed neutron group, and it is evaluated depending on feedback temperature dynamics [2]. The following ARX scheme [3] represents the system model in the quasi-linear discrete form driven by closed loop pole locations -0.1, -0.2, and -0.3 and it is given by:

$$y_{(n+1)} = -t_1 \cdot y_{(n)} - t_2 \cdot y_{(n-1)} - t_3 \cdot y_{(n-2)} + h \cdot (b_0 \cdot r_{(n)} + b_1 \cdot r_{(n-1)} + b_2 \cdot r_{(n-2)}) + \varepsilon$$

Parameters of the closed loop model are listed in table (1). The value of the compensator parameter  $h$  can be calculated from the relation  $h = (1+t_1+t_2+t_3)/(b_0+b_1+b_2)$ .

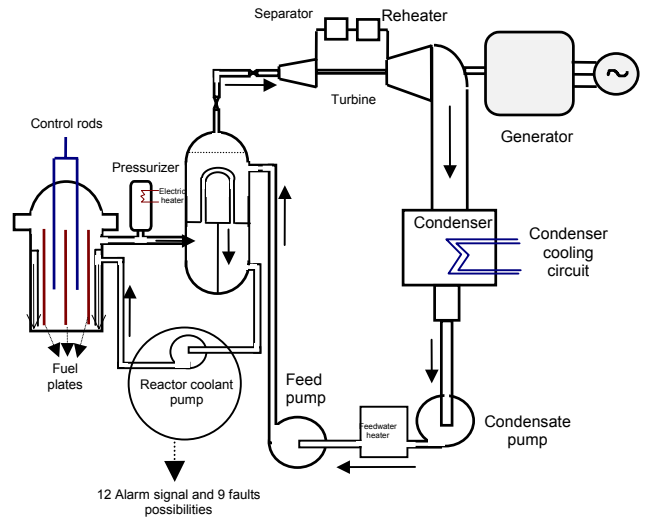


Fig. (1) Nuclear Power Station

$b_0 = 0.99882$	$b_1 = -0.868$	$b_2 = -0.06618$
$t_1 = 0.6$	$t_2 = 0.11$	$t_3 = 0.006$

Table (1) Quasi-linear discrete model parameters  $B$ , and  $T$

### III. ERROR BACK PROPAGATION TRAINING ALGORITHM

The MNN has three layers input, hidden, and output. The parameters of the learning rule are as follow:

- $I...$  is the number of inputs nodes
- $J...$  is the number of hidden nodes
- $K...$  is the number of outputs nodes
- $P...$  is the number of patterns
- $X...$  is the input,  $h...$  is the hidden layer output
- $O...$  is the MNN output,  $w...$  is the output weights
- $v...$  is the hidden layer weights
- $d...$  is the desired,  $\eta...$  is the learning speed constant
- $RMS...$  Root Mean Square of error

The EBPTA algorithm [4] can be accomplished as follows:

**Step 1:**  $\eta > 0, E_{\max}$  chosen.

Weights  $W$  and  $V$  are initialized at a small random values,  $W$  is  $(K \times J)$  and  $V$  is  $(J \times I)$ .

**Step 2 :** Training step starts here. The input is presented and the layer's output is computed.

$$h_j = f(v_j^t x), \text{ for } j = 1, 2, 3, \dots, J$$

$$o_k = f(w_k^t h), \text{ for } k = 1, 2, 3, \dots, K$$

**Step 3 :** Error Value is computed;

$$E = \frac{1}{2} (d_k - o_k)^2 + E$$

**Step 4 :** Error signal vectors  $\Delta_o$  and  $\Delta_y$  of both output and hidden layer are computed. Vector  $D_o$  is  $(K \times 1)$  and  $D_y$  is  $(J \times 1)$ . The error signal term of output layer is

$$\delta_{ok} = \frac{1}{2} (d_k - o_k)(1 - o_k^2), \text{ for } k = 1, 2, \dots, K$$

The error signal term of the hidden layer in this step

$$\delta_{yj} = (1 - y_j) \sum_{k=1}^K \delta_{ok} W_{kj}, \text{ for } j = 1, 2, \dots, J$$

**Step 5:** Output layer weights will be adjusted as:

$$w_{kj} \leftarrow w_{kj} + \eta \delta_{ok} y_j,$$

for  $k = 1, 2, \dots, K$  and  $j = 1, 2, \dots, J$

**Step 6:** Hidden layer weights will be adjusted as:

$$v_{ji} \leftarrow v_{ji} + \eta \delta_{yj} z_i,$$

for  $j = 1, 2, \dots, J$  and  $i = 1, 2, \dots, I$

**Step 7:** If  $p < P$  then  $p \leftarrow p+1$ , and go to step 2; otherwise, go to step 8.

**Step 8:** The training cycle is completed. For  $E < E_{\max}$  terminate the training session. Output weights  $W, V$ , and  $E$ . If  $E < E_{\max}$ , then  $E \leftarrow 0, p \leftarrow 1$ , and initiate the new training cycle by going to step 2.

### IV. THE RESULTS OF SIMULATOR LEARNING

The design of the MNN simulator of NPR [5-6] has 6 nodes in the input layer then  $I=6$ , 7 nodes in the hidden layer then  $J=7$ , and single output node then  $K=1$ . The ANNs simulator diagram of NPR is shown in fig.(2).

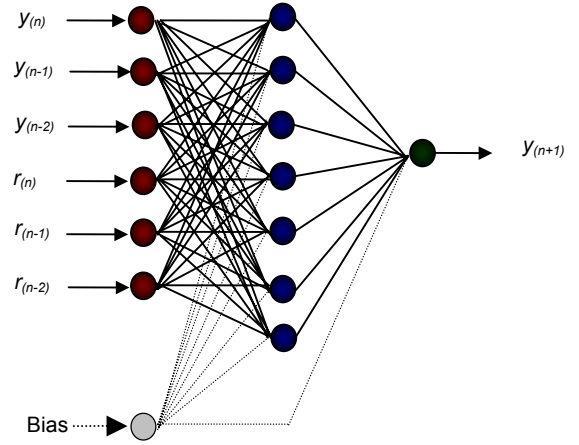


Fig. (2) MNN simulator of NPR.

The training parameters used to learn the ANNs simulator of the feedback temperature model of the NPR are shown in the table (2).

Parameters of Network Learning	Parameter value
Learning speed constant ( $c$ )	0.2
Momentum term constant ( $\alpha$ )	0.5
Activation function constant ( $\lambda$ )	1
Number of layers	3
Size of input layer	6
Size of hidden layer	7
Size of output layer	1
Bias	1
Total weights	57
Learning iterations	885
Good patterns percent %	100%
Target error	0.001
Number of patterns	18

Table (2) MNN simulator parameters of NPR.

The system output  $Y_{\text{sys}}$  and the simulator output  $Y_{\text{sim}}$  with respect to reference  $R$  are shown in the fig. (3). This figure (3) shows how the simulator output is closely typical to system output. The convergence of the output of the hidden layer nodes are shown in fig. (4) w.r.t epoch index. The decrement of the  $RMS$  error while the learning process is shown in fig. (5). When the  $RMS$  value reaches the desired value, the learning process stops. The training pattern error is shown in fig. (6), this illustrates how the training error changes w.r.t the system output. After learning, the ANNs weights get their final values. The

obtained weights are the main ANNs simulator parameters. These values are listed in the table (3.a,b).

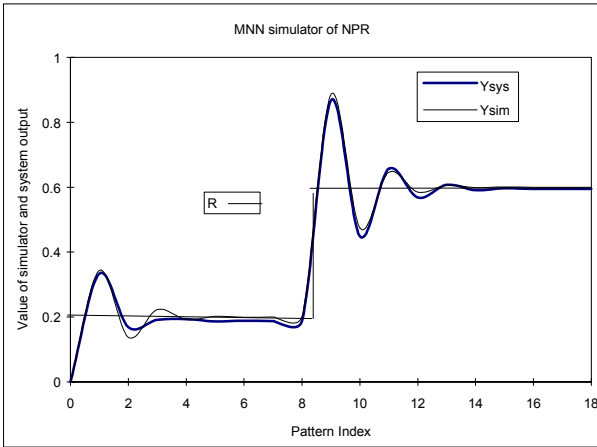


Fig.(3) NPR output  $Y_{sys}$  compared with simulator output  $Y_{sim}$  following reference  $R$ .

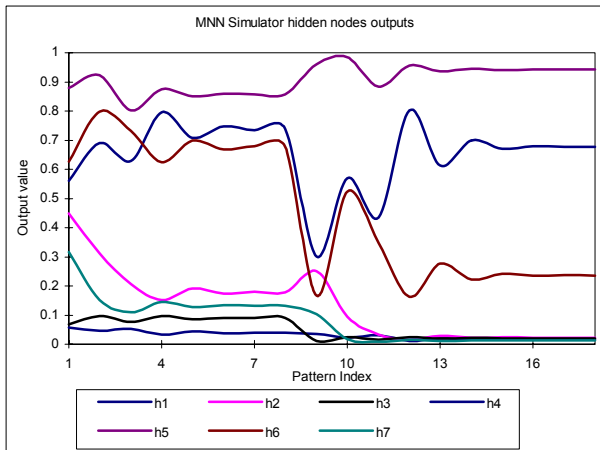


Fig. (4) MNN simulator hidden layer nodes outputs

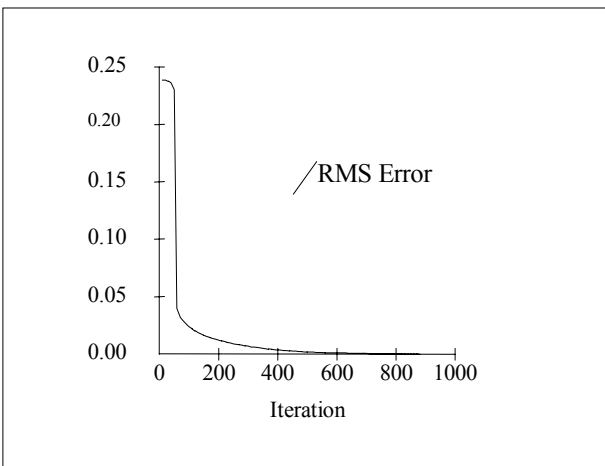


Fig.(5) RMS decrement during the learning.

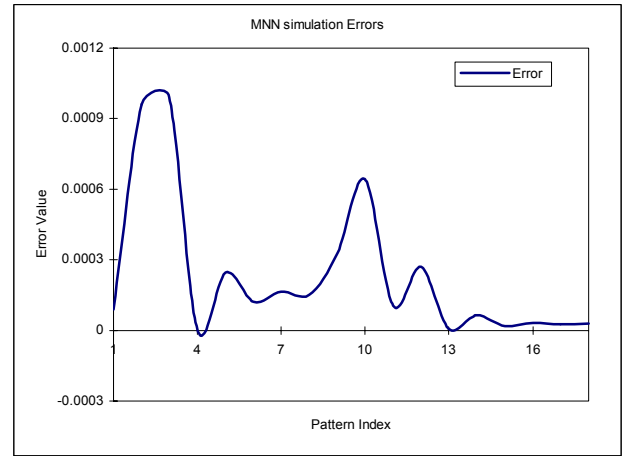


Fig.(6) Simulator training patterns error.

## V. ENHANCEMENT OF SIMULATOR LEARNING

To speedup the learning process, we have used the momentum method. Changing ANNs weights based on the MNN learning rule yields the following rule:

$$W(t+1) = W(t) + \eta r[w(t), X(t), d(t)]X(t) + \alpha (w(t) - w(t-1))$$

Whereas;

$r$ ... Learning signal                       $W$ ... Weights vector  
 $\alpha$  .. momentum constant                 $d$ ... desired output  
 $\eta$  ... Learning speed constant         $X$ ... Net. input

To enhance the speed and the precision of the simulator learning process, the learning speed constant  $\eta$  and the momentum constant  $\alpha$  have to be optimized. These values can be estimated by learning of learning (correlation technique), which leads to speeding up the learning process. If these parameters go out of these defined values, the simulator will not give reasonable results. Fig.(7) shows the effect of changing the parameters  $\eta$  and  $\alpha$  on the learning speed. The main processing function of non-linear ANNs is the sigmoidal function, which is given by the following form:

$$f = \frac{1}{1 + \exp(-\lambda \cdot \text{net})}$$

The change of activation function constant  $\lambda$  affects the simulator learning by the same way of  $\eta$  parameter. The effect of changing of activation function constant  $\lambda$  on the learning speed is shown in fig. (8).

Input/Hidden	$I_1$	$I_2$	$I_3$	$I_4$	$I_5$	$I_6$	Bias
$J_1$	-0.708454	0.075490	-1.341838	-2.582520	1.615173	0.619263	-2.728786
$J_2$	-1.702557	-2.652719	-2.278315	1.627022	1.000254	-1.548759	-0.419965
$J_3$	1.104663	-0.059526	0.453599	-1.866894	-2.431957	-1.035228	-1.549520
$J_4$	1.630167	0.191998	2.134559	-1.524039	-1.597829	-1.555349	1.174401
$J_5$	1.446251	-2.283529	-0.173770	0.631515	2.441326	0.443648	1.286075
$J_6$	2.485758	0.432553	-1.813839	-1.928107	-0.964793	-2.990123	1.696176
$J_7$	-2.768206	-2.759605	-0.041310	2.381305	0.171234	-3.238385	-0.631515

Table (3.a) Hidden layer weights patterns.

$J_1$	$J_2$	$J_3$	$J_4$	$J_5$	$J_6$	$J_7$	Bias
0.739920	0.145607	1.377910	-3.239505	1.905194	-3.586458	-0.107429	1.614967

Table (3.b) Output layer weights pattern.

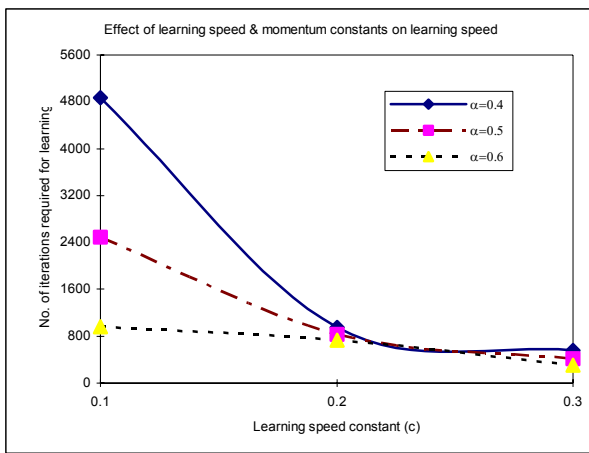


Fig.(7) Effect of changing Learning speed constant ( $\eta$ ) and Momentum term  $\alpha$  on learning speed

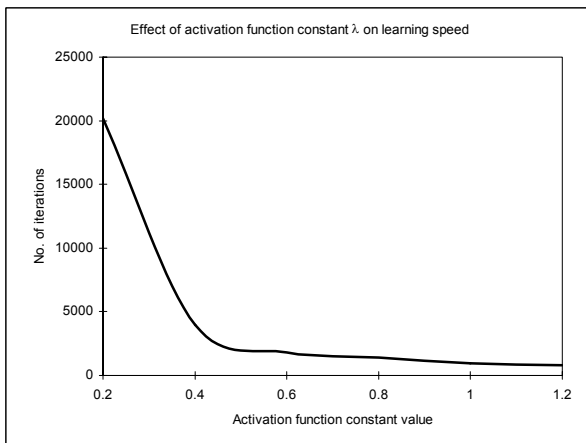


Fig.(8) Effect of activation function constant  $\lambda$  on learning speed

From the presented results we can notice that the smooth decrement of *RMS* error fig.(5) indicates that, the learning parameters, shown in table (2), have right values and the learning process is successful. The maximum value of patterns error is 0.01, shown in fig.(6). This

indicates that ANNs simulator has a high precision. The correlation between the reference output patterns and the output of the ANNs tend to be 1, shown in fig.(3). Hence, the ANNs simulator is well trained and can easily simulate the quasi-linear NPR model.

## VI. CONCLUSION

The previous results illustrate, how the MNN can be learned to simulate the reactor behaviour with a high precision in quasi-linear or even non-linear modes, shown in fig.(3). Moreover, using the momentum method is a successful technique, which enhances the precision and the speed of the learning process. Based on the same technique, ANNs can be used to model and to simulate several dynamics of the NPR or another system with a high precision and reliability.

## REFERENCES:

- [1] Wu Jian-Kang, "Neural Networks and Simulation Methods", Marcel Dekker Inc., 1994.
- [2] R.M. Edward, "Robust Optimal Control of Nuclear Reactors", Ph.D. thesis, Dep. of nuclear engineering, Pen. state university, May 1990.
- [3] Jack Golten and Andy Verwer, "Control System Design and Simulation", McGraw-Hill, 1991.
- [4] J. M. Zurada, "Introduction to Artificial Neural Systems", West Publishing Co., 1992.
- [5] Ku, Chao-chee, K.Y. Lee, and R.M. Edward, "Improved Nuclear Reactor Temperature Control using Diagonal Recurrent Neural Networks", IEEE transaction on nuclear science, 39: pp(2298-2308), Des. 1992.
- [6] K.S.Narendran and K.Parthasarathy, "Identification and Control of Dynamical Systems using Neural Networks", IEEE transaction on Neural Networks., vol. 1, No. 1, March. 1990.