Hindawi Publishing Corporation
Science and Technology of Nuclear Installations
Volume 2008, Article ID 681890, 10 pages
doi:10.1155/2008/681890

*Research Article*

# Validation of Infinite Impulse Response Multilayer Perceptron for Modelling Nuclear Dynamics

**F. Cadini, E. Zio, and N. Pedroni**

*Department of Nuclear Engineering, Polytechnic of Milan, Via Ponzio 34/3, 20133 Milan, Italy*

Correspondence should be addressed to F. Cadini, francesco.cadini@polimi.it

Received 2 May 2007; Revised 16 November 2007; Accepted 3 December 2007

Recommended by Nikola Cavlina

Artificial neural networks are powerful algorithms for constructing nonlinear empirical models from operational data. Their use is becoming increasingly popular in the complex modeling tasks required by diagnostic, safety, and control applications in complex technologies such as those employed in the nuclear industry. In this paper, the nonlinear modeling capabilities of an infinite impulse response multilayer perceptron (IIR-MLP) for nuclear dynamics are considered in comparison to static modeling by a finite impulse response multilayer perceptron (FIR-MLP) and a conventional static MLP. The comparison is made with respect to the nonlinear dynamics of a nuclear reactor as investigated by IIR-MLP in a previous paper. The superior performance of the locally recurrent scheme is demonstrated.



## 1. INTRODUCTION

Several design and verification activities in the field of nuclear power plant engineering rely on the simulation of the plant dynamic response under different scenarios and conditions. However, the complexity and nonlinearities of the involved processes are such that analytical modelling becomes burdensome, if at all feasible.

For this reason, empirical modelling is becoming very popular since it does not require a detailed physical understanding of the processes or knowledge of the material properties, geometry, and other characteristics of the plant and its components. The underlying dynamic model is identified by fitting plant operational data with a procedure often referred to as "learning."

In this respect, artificial neural networks are powerful algorithms for constructing nonlinear empirical models from operational data. As a fact, artificial neural networks are being used with increasing frequency as an alternative to traditional models in a variety of engineering applications including monitoring, prediction, diagnostics, control, and safety.

Whereas standard feedforward neural networks can model only static input/output mappings [1–4], recurrent neural networks (RNNs) have been proven to be universal approximators of nonlinear dynamic systems [5–7].

Two main methods exist for providing a neural network with dynamic behavior, that is, the insertion of a buffer somewhere in the network to provide an explicit memory of the past inputs, or the implementation of feedbacks.

As for the first method, it builds on the structure of feedforward networks where all input signals flow in one direction, from input to output. Since a feedforward network does not have a dynamic memory, *tapped delay lines* (temporal buffers) of the inputs are introduced. The buffers can be applied at the network inputs only, keeping the network internally static as in the buffered multilayer perceptron (MLP) [8] or at the input of each neuron as in the MLP with finite impulse response filter synapses (FIR-MLP) [9, 10]. The main disadvantage of the buffer approach is the limited past history horizon which needs to be used in order to keep the size of the network computationally manageable, thereby preventing modelling of arbitrary long-time dependencies between inputs and outputs [11]. It is also difficult to set the length of the buffer given a certain application.

Regarding the second method, the most general example of implementation of feedbacks in a neural network is the fully recurrent neural network constituted by a single layer of neurons fully interconnected with each other [12] or by several such layers [13, 14]. Because of the required large structural complexity of this network, in recent years

growing efforts have been propounded in developing methods for implementing temporal dynamic feedback connections into the widely used multilayered feedforward neural networks. Recurrent connections can be added by using two main types of recurrence or feedback: *external* or *internal*. *External recurrence* is obtained, for example, by feeding back the outputs to the input of the network, as in NARX networks [15–18]; *internal recurrence* is obtained by feeding back the outputs of neurons of a given layer to inputs of neurons of the same layer, giving rise to the so-called *locally recurrent neural networks* (*LRNNs*) [19, 20].

The major advantages of LRNNs with respect to the buffered tapped delayed feedforward networks and to the fully recurrent networks are [19] as follows: (1) the hierarchic multilayer topology on which they are based is well known and efficient; (2) the use of dynamic neurons allows to limit the number of neurons required for modelling a given dynamic system, contrary to the tapped delayed networks; (3) the training procedures for properly adjusting the network weights are significantly simpler and faster than those for the fully recurrent networks.

In a previous paper [21], an infinite impulse response locally recurrent neural network (IIR-LRNN) has been trained by a recursive backpropagation (RBP) algorithm [19] to track the nonlinear continuous time dynamics of a nuclear reactor [22]. In the IIR-LRNN, the synapses are implemented as infinite impulse response (IIR) digital filters, which provide the network with system state memory.

In this paper, the same case study is considered to show the benefits gained from the use of the IIR-LRNN by making similar comparisons as in [19] with two static networks, namely, an FIR-MLP and a conventional static MLP.

The paper is organized as follows. For completeness and self-consistency, in Section 2, the main features of the IIR-LRNN architecture and forward calculation are briefly summarized [19]. In Section 3, the application of the IIR-LRNN to the reactor neutron flux dynamics made in [21] is illustrated and then compared to that of the two above mentioned static neural models. The conclusions drawn from such comparison are proposed in Section 4.

## 2. LOCALLY RECURRENT NEURAL NETWORKS

### 2.1. The IIR-LRNN architecture and forward calculation

The following description is a brief synthesis of the illustration of the IIR-LRNN given in [19, 21].

An LRNN is a time-discrete network consisting of a global feedforward structure of nodes interconnected by synapses which link the nodes of the $k$th layer to those of the successive $(k+1)$th layer, $k = 0, 1, \ldots, M$, with layer 0 being the input and $M$ being the output. Different from the classical static feedforward networks, in an LRNN, each synapse carries taps and feedback connections. In particular, each synapse of an IIR-LRNN contains an IIR linear filter whose characteristic transfer function can be expressed as ratio of two polynomials with poles and zeros representing the autoregressive (AR) and moving average (MA) parts of the model, respectively.

During the forward phase, at the generic time $t = 1, 2, \ldots, T$, the generic neuron $j = 1, 2, \ldots, N^k$ belonging to the generic layer $k = 0, 1, \ldots, M$ receives in input the quantity $y_{jl}^k(t)$ from neuron $l = 1, 2, \ldots, N^{k-1}$ of layer $(k-1)$:

$$y_{jl}^k(t) = \sum_{p=0}^{L_{jl}^k - 1} w_{jl(p)}^k \cdot x_l^{k-1}(t-p) + \sum_{p=1}^{I_{jl}^k} v_{jl(p)}^k \cdot y_{jl}^k(t-p). \tag{1}$$

The quantities $y_{jl}^k(t)$, $l = 1, 2, \ldots, N^{k-1}$, are summed to obtain the net input $s_j^k(t)$ to the nonlinear activation function $f^k(\cdot)$, which is typically a sigmoidal Fermi function of the $j$th node, $j = 1, 2, \ldots, N^k$, of the $k$th layer, $k = 1, 2, \ldots, M$:

$$s_j^k(t) = \sum_{l=0}^{N^{k-1}} y_{jl}^k(t). \tag{2}$$

The output of the activation function gives the state of the $j$th neuron of the $k$th layer $x_j^k(t)$:

$$x_j^k(t) = f^k\left[s_j^k(t)\right] \quad (= 1 \text{ for the bias node, } j = 0). \tag{3}$$

For simplicity of illustration, and with no loss of generality, an example of a network constituted by only one hidden layer (i.e., $M = 2$) is depicted in Figure 1.

Note that if all the synapses contain only the MA part (i.e., $I_{jl}^k = 0$ for all $j, k, l$), the architecture reduces to an FIR-LRNN, and if all the synaptic filters contain no memory (i.e., $L_{jl}^k - 1 = 0$ and $I_{jl}^k = 0$ for all $j, k, l$), the classical multilayered feedforward static neural network is obtained.

Further details about the IIR-LRNN architecture and forward calculation may be found in [19, 21].

### 2.2. The recursive backpropagation (RBP) algorithm for batch training

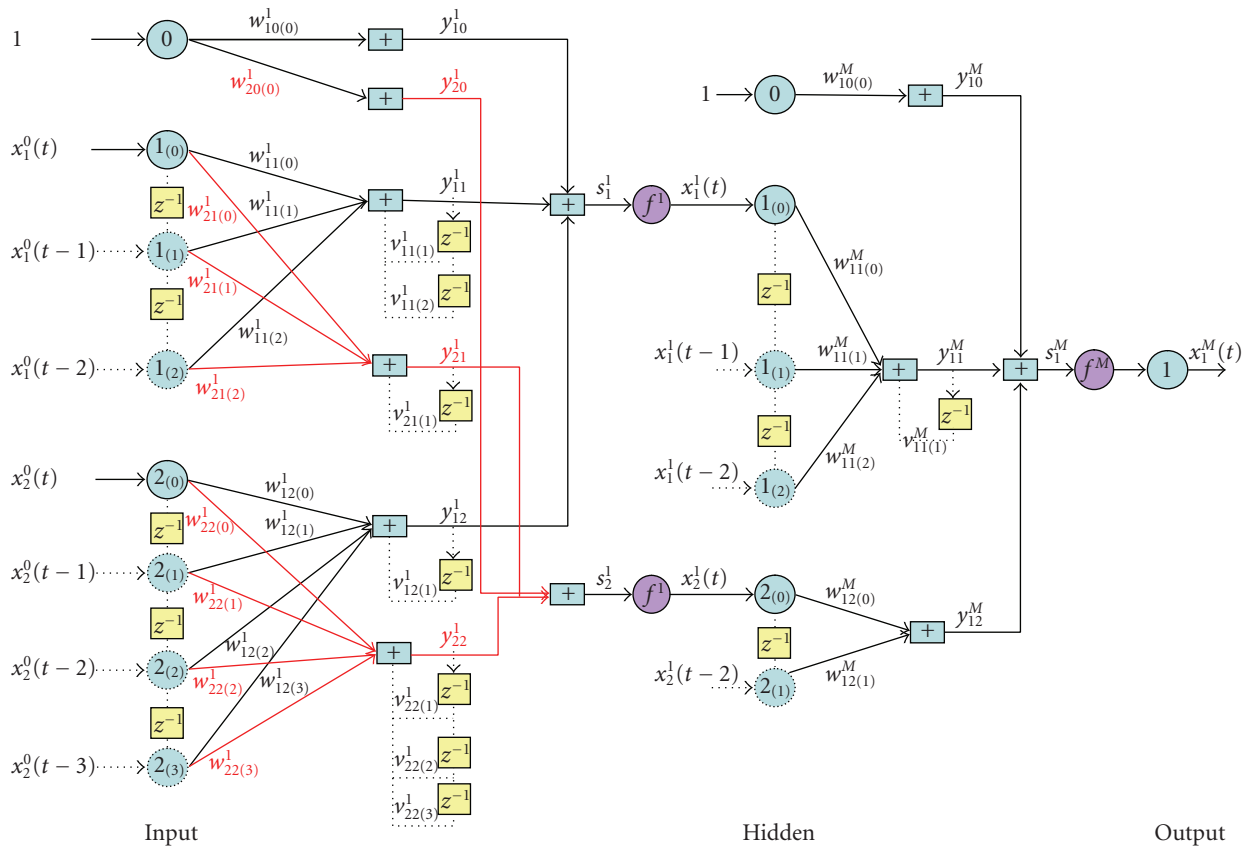
The recursive backpropagation (RBP) training algorithm [19] is a gradient-based minimization algorithm which makes use of a particular chain rule expansion for the computation of the necessary derivatives. When used in batch mode, it is equivalent to real-time recurrent learning (RTRL) [23] and backpropagation through time (BPTT) [24]. For brevity, the RBP algorithm is not presented in this paper; the interested reader may refer to [19, 21] for details.

## 3. THE LRNN MODEL FOR THE SIMULATION OF NEUTRON FLUX DYNAMICS

In this section, we first proceed to illustrate the case study and its development by the IIR-LRNN presented in [21] (see Sections 3.1 and 3.2), then we proceed to the comparison of the achieved performance with two static neural networks, namely, an FIR-MLP and a conventional static MLP properly devised to the scope (see Section 3.3).

### 3.1. The dynamic system

The neutron flux dynamics are described by a simple model based on a one-group point kinetics equation with nonlinear




| Input ($k = 0$) | Hidden ($k = 1$) | Output ($k = 2 = M$) |
|---|---|---|
| $N^0 = 2$ (nodes) | $N^1 = 2$ (nodes) | $N^M = 1$ (nodes) |
| | $L^1_{11} - 1 = 2$, MA-synapses (taps), $w^1_{11(p)}$, $w^1_{21(p)}$<br>$L^1_{12} - 1 = 3$, MA-synapses (taps), $w^1_{12(p)}$, $w^1_{22(p)}$ | $L^M_{11} - 1 = 2$, MA-synapses (taps), $w^M_{11(p)}$<br>$L^M_{12} - 1 = 1$, MA-synapses (taps), $w^M_{12(p)}$ |
| | $I^1_{11} = 2$, AR-synapses (feedback), $v^1_{11(p)}$, $v^1_{21(p)}$<br>$I^1_{12} = 2$, AR-synapses (feedback), $v^1_{12(p)}$, $v^1_{22(p)}$ | $I^M_{11} = 1$, AR-synapses (feedback), $v^M_{11(p)}$<br>$I^M_{12} = 0$, AR-synapses (feedback), $v^M_{12(p)}$ |

Figure 1: Scheme of an IIR-LRNN with one hidden layer.

power reactivity feedback, combined with xenon and iodine balance equations [22]:

$$\begin{aligned}
\Lambda\frac{d\Phi(t)}{dt} &= \left[(\rho_0 + \Delta\rho(t)) - \frac{\sigma_{Xe}}{c\Sigma_f}Xe(t) - \gamma\Phi(t)\right]\Phi(t),\\
\frac{dXe(t)}{dt} &= \gamma_{Xe}\Sigma_f\Phi(t) + \lambda_I I(t) - \lambda_{Xe}Xe(t) - \sigma_{Xe}Xe(t)\Phi(t),\\
\frac{dI(t)}{dt} &= \gamma_I\Sigma_f\Phi(t) - \lambda_I I(t),
\end{aligned} \tag{4}$$

with the usual nuclear physics meaning of the symbols employed (see Acronyms and Symbols).

The reactor evolution is assumed to start from an equilibrium state at a nominal flux level $\Phi_0 = 4.66\cdot10^{12}$ n/cm$^2$ s. The initial reactivity needed to keep the steady state is $\rho_0 = 0.071$, and the xenon and iodine concentrations are $Xe_0 = 5.73\cdot10^{15}$ nuclei/cm$^3$ and $I_0 = 5.81\cdot10^{15}$ nuclei/cm$^3$, respectively. In what follows, the values of flux, xenon, and iodine concentrations are normalized with respect to these steady-state values.

The objective is to design and train an LRNN to reproduce the neutron flux dynamics described by the system of differential equations (see (4)), that is, to estimate the evolution of the normalized neutron flux $\Phi(t)$, knowing the forcing function $\rho(t)$.

Notice that the estimation is based only on the current values of reactivity. These are fed in input to the locally recurrent model at each time step $t$. Thanks to the MA and AR parts of the synaptic filters, an estimate of the neutron flux $\hat{\Phi}(t)$ at time $t$ is produced, which recurrently accounts for past values of both the network inputs and the estimated outputs, namely,

$$\hat{\Phi}(t) = F(\rho(t), \rho(t-1), \ldots, \hat{\Phi}(t-1), \hat{\Phi}(t-2), \ldots, \boldsymbol{\Theta}), \tag{5}$$

Table 1: Training parameters of the LRNN for simulating the reactor neutron flux.

| Principal training parameters | |
|---|---|
| Number of transients in the training set | 250 |
| Number of patterns in each transient | 50 |
| $\mu$ (learning coefficient) | 0.001 |
| $\alpha$ (momentum coefficient) | 0 |
| Learning epochs ($n_{\text{epoch}}$) | 200 |
| Consecutive repetitions of each transient ($n_{\text{rep}}$) | 10 |
| Data normalization range | 0.2–0.8 |

where $\boldsymbol{\Theta}$ is the set of adjustable parameters of the network model, that is, the synaptic weights.

On the contrary, the other nonmeasurable system state variables, $Xe(t)$ and $I(t)$, are not fed in input to the LRNN; the associated information remains distributed in the hidden layers and connections, which renders the LRNN modelling task quite difficult.

### 3.2. The LRNN training

The LRNN used in this work is characterized by three layers: the input layer with two nodes (bias included), the hidden layer with six nodes (bias included), and the output layer with one node. A sigmoidal activation function has been adopted for the hidden and output nodes.

The training set is made up of $N_t = 250$ transients, with each one lasting for $T = 2000$ minutes and sampled with a time step $\Delta t$ of 40 minutes, thus generating $n_p = 50$ patterns. Notice that a temporal length of 2000 minutes allows for the development of the long-term dynamics which are affected by the long-term $Xe$ oscillations.

All data have been normalized in the range of 0.2–0.8.

Each transient has been created varying the reactivity from its steady-state value according to the following step function:

$$\rho(t) = \begin{cases} \rho_0, & t \le T_s, \\ \rho_0 + \Delta\rho, & t > T_s, \end{cases} \tag{6}$$

where $T_s$ is a random steady-state time interval and $\Delta\rho$ is a random reactivity variation amplitude. In order to build the 250 different transients for the training, these two parameters have been randomly chosen within the ranges of 0–2000 minutes and $-5\cdot10^{-4} + 5\cdot10^{-4}$, respectively.

The training procedure has been carried out on the available data for $n_{\text{epoch}} = 200$ learning epochs (iterations). During each epoch, every transient is repeatedly presented to the LRNN for $n_{\text{rep}} = 10$ consecutive times. The weight updates are performed in batch at the end of each training sequence of length $T$. Neither momentum term nor an adaptive learning rate [19] turned out to be necessary for increasing the efficiency of the training in this case. The principal training parameters are summarized in Table 1.

The number of delays (orders of the MA and AR parts of the synaptic filters) has been set by trial-and-error, so as to obtain a satisfactory performance of the LRNN, measured in terms of a small root mean square error (RMSE) on the training set. The best LRNN structure resulting from these tests is summarized in Table 2.

Table 2: Structure of the LRNN for the simulation of the reactor neutron flux.

| LRNN structure | | |
|---|---|---|
| Input nodes (bias included) | | 2 |
| Hidden nodes (bias included) | | 6 |
| Output nodes | | 1 |
| Type of activation functions (hidden/output nodes) | | Sigmoidal |
| $L^k_{jl}$ | Hidden MA ($k = 1, l = 1, j = 1, 2,\ldots, 5$) | 12 |
| | Output MA ($k = M = 2, l = 1, 2,\ldots, 5, j = 1$) | 12 |
| $I^k_{jl}$ | Hidden AR ($k = 1, l = 1, j = 1, 2,\ldots, 5$) | 10 |
| | Output AR ($k = M = 2, l = 1, 2, \ldots, 5, j = 1$) | 10 |

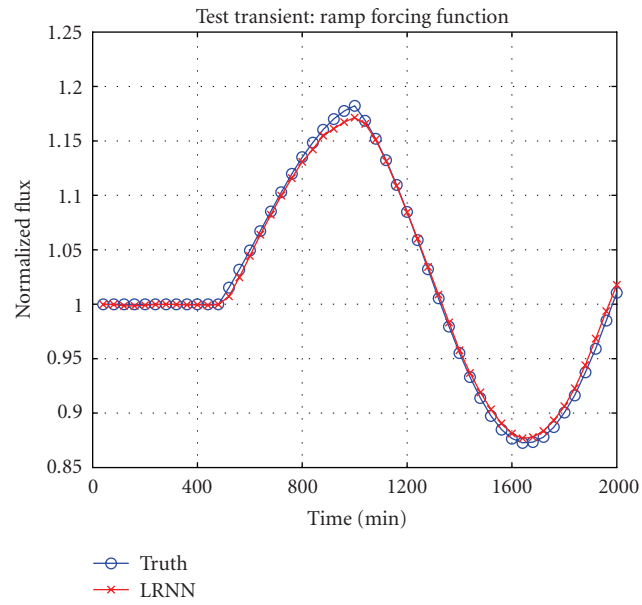



Figure 2: Comparison of the model-simulated normalized flux (circles) with the LRNN-estimated one (crosses), for one sample ramp transient of the test set.

### 3.3. Results

The generalization capability of the LRNN is verified on test transients generated by forcing functions' variations quite different from those used in the training phase (e.g., ramp, sinusoidal, and random variations).

The evolutions of the flux, normalized with respect to the steady-state value $\Phi_0$, corresponding to three sample transients of the test set are reported in Figures 2, 3, and 4. The LRNN estimate of the output (crosses) is in satisfactory agreement with the actual transient (circles), even for dynamics quite different from those used for the network training. Notice the ability of the LRNN to deal with both the short-term dynamics governed by the instantaneous variations of the forcing function (i.e., the reactivity step) and the long-term dynamics governed by $Xe$ oscillations.

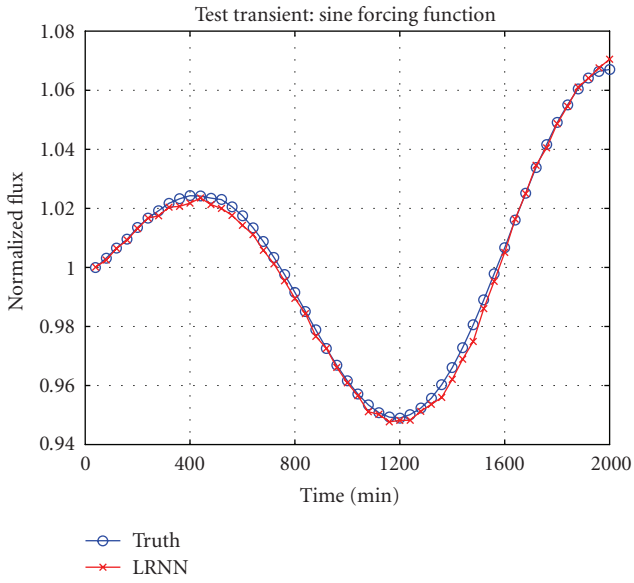



Figure 3: Comparison of the model-simulated normalized flux (circles) with the LRNN-estimated one (crosses), for one sample sinusoidal transient of the test set.

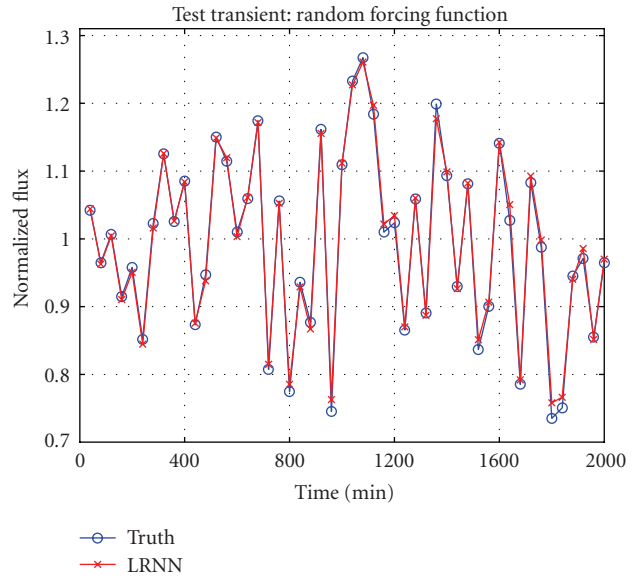



Figure 4: Comparison of the model-simulated normalized flux (circles) with the LRNN-estimated one (crosses), for one sample random transient of the test set.

Furthermore, the computing time is about 5000 times lower than that required by the numerical solution of the underlying model (4). This makes the LRNN model very attractive for real-time applications, for example, for control or diagnostic purposes, and applications for which repeated evaluations are required (e.g., uncertainty and sensitivity analyses).

### 3.3.1. Comparison with two static neural networks

Two additional static neural network models have been examined for comparison: a buffered multilayer perceptron

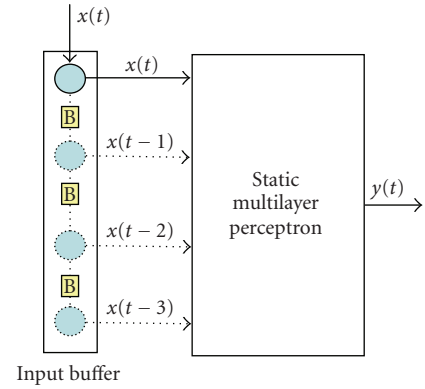



Figure 5: Example of buffered MLP with input buffer.

(MLP), where *tapped delay lines* are applied at the network inputs only, keeping the network internally static (see Figure 5) [8], and a finite impulse response multilayer perceptron (FIR-MLP), where temporal buffers are applied at the input of each neuron; that is, all connection weights are realized by linear FIR filters (see Figure 6) [8, 10, 25–27]. In passing, notice that the buffered MLP and FIR-MLP can be shown to be theoretically equivalent since the internal buffers can be implemented as an external one [27]. However, to implement an FIR-MLP as a buffered MLP, the first layers' subnetworks must be replicated with shared weights, and this increases the complexity with respect to the case of considering the internal buffer [27]. This leads to different architectures of the buffered MLP and FIR-MLP in their actual implementations.

For a fair comparison, the structures of the static neural networks considered have been selected so that they contain approximately the same number of adaptable parameters as does the IIR-LRNN described in Section 3.2. In particular, the buffered MLP is chosen with fourteen hidden neurons (bias included) and fifteen input delays, whereas the FIR-MLP is selected with ten hidden neurons (bias included) and linear FIR filters of twelfth order.

Three different learning algorithms have been used: standard static backpropagation (BP) for the buffered MLP, temporal backpropagation (TBP) for the FIR-MLP, and recursive backpropagation (RBP) for the IIR-LRNN. The information concerning the structures and learning algorithms of the three neural networks is summarized in Table 3.

The training procedures have been carried out on the learning dataset described in Section 3.2, and their results have been expressed in terms of the root mean square error (RMSE) computed after each learning epoch. From Figure 7, it is evident that the IIR-LRNN outperforms both the static MLP and the FIR-MLP, showing better modelling capabilities, faster training, and significantly higher accuracy; the asymptotic RMSE values are 0.081 for the static MLP, 0.075 for the FIR-MLP, and 0.007 for the IIR-LRNN.

The representation and generalization capabilities of the three neural architectures considered have then been compared on a number of different test datasets. The results are synthesized in Table 4 in terms of root mean square error

Table 3: Structures and learning algorithms of the three neural networks involved in the comparison (i.e., buffered MLP, FIR-MLP, and IIR-LRNN).

| Neural architecture | Network structure | Learning algorithm |
|---|---|---|
| Buffered MLP | Hidden nodes: 14 (bias included)<br>Input delays: 15<br>Feedback delays: 0 | Static backpropagation (BP) |
| FIR-MLP | Hidden nodes: 10 (bias included)<br>Hidden MA-AR: 12-0<br>Output MA-AR: 12-0 | Temporal backpropagation (TBP) |
| IIR-LRNN | Hidden nodes: 6 (bias included)<br>Hidden MA-AR: 12-10<br>Output MA-AR: 12-10 | Recursive backpropagation (RBP) |

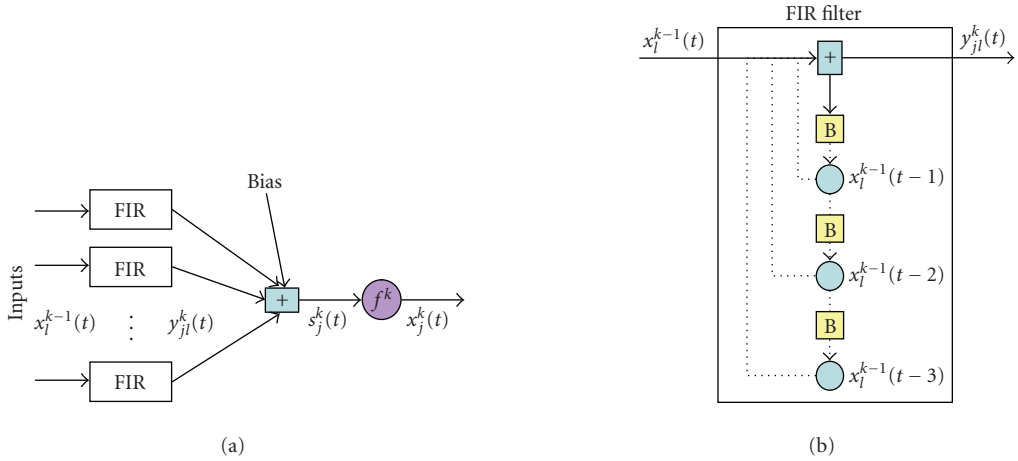



Figure 6: (a) Model of the neuron for an FIR-MLP and (b) example of an FIR filter of fourth order.

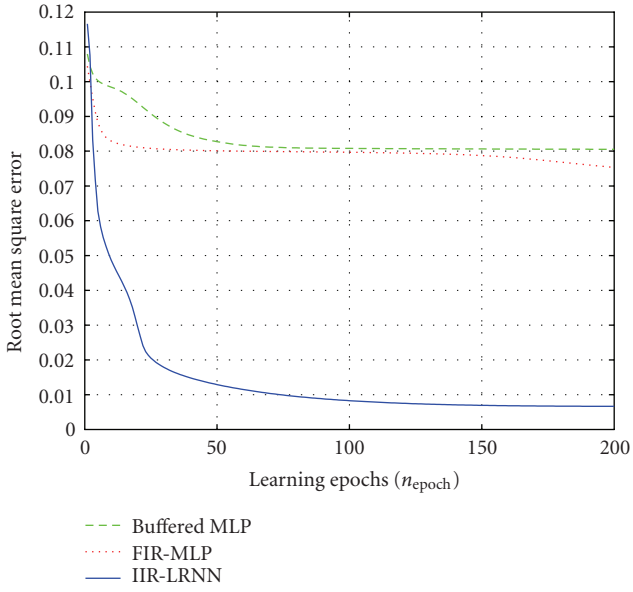



Figure 7: Convergence performance of the buffered MLP, FIR-MLP, and IIR-LRNN applied to the reactor neutron flux estimation.

(RMSE), mean absolute error (MAE), and mean relative error (MRE):

$$\text{RMSE} = \sqrt{\frac{1}{N_t \cdot n_p} \cdot \sum_{n=1}^{N_t} \sum_{o=1}^{n_p} \left(\hat{\Phi}_{n,o} - \Phi_{n,o}\right)^2},$$

$$\text{MAE} = \frac{1}{N_t \cdot n_p} \cdot \sum_{n=1}^{N_t} \sum_{o=1}^{n_p} \left|\hat{\Phi}_{n,o} - \Phi_{n,o}\right|, \tag{7}$$

$$\text{MRE} = \frac{1}{N_t \cdot n_p} \cdot \sum_{n=1}^{N_t} \sum_{o=1}^{n_p} \frac{\hat{\Phi}_{n,o} - \Phi_{n,o}}{\Phi_{n,o}},$$

where $\Phi_{n,o}$ and $\hat{\Phi}_{n,o}$ are the values of the normalized flux and its neural estimate in the $o$th pattern of the $n$th transient, respectively.

Owing to the richness of the network architecture, the IIR-LRNN model exhibits consistently better performance compared to the static models. For instance, considering the ramp test set of Figure 2, the IIR-LRNN for the normalized neutron flux provides an RMSE of 0.0049 and an MAE

Table 4: Values of the performance indices (RMSE, MAE, MRE) calculated over different test sets for the buffered MLP, FIR-MLP, and IIR-LRNN trained to estimate the reactor neutron flux dynamics.

| Buffered MLP | | | | |
|---|---|---|---|---|
| | | | Errors | |
| Forcing function | Number of sequences | RMSE | MAE | MRE |
| Step | 80 | 0.0805 | 0.0523 | 0.0051 |
| Ramp | 80 | 0.0751 | 0.0503 | 0.0065 |
| Sine | 80 | 0.0763 | 0.0513 | −0.0041 |
| Random | 80 | 0.0792 | 0.0520 | 0.0047 |
| FIR-MLP | | | | |
| | | | Errors | |
| Forcing function | Number of sequences | RMSE | MAE | MRE |
| Step | 80 | 0.0753 | 0.0487 | 0.0024 |
| Ramp | 80 | 0.0698 | 0.0435 | 0.0050 |
| Sine | 80 | 0.0705 | 0.0450 | −0.0035 |
| Random | 80 | 0.0748 | 0.0482 | 0.0038 |
| IIR-LRNN | | | | |
| | | | Errors | |
| Forcing function | Number of sequences | RMSE | MAE | MRE |
| Validation Step | 80 | 0.0098 | 0.0060 | −0.0003 |
| Ramp | 80 | 0.0049 | 0.0039 | 0.0037 |
| Sine | 80 | 0.0058 | 0.0051 | −0.0011 |
| Random | 80 | 0.0063 | 0.0054 | 0.0021 |

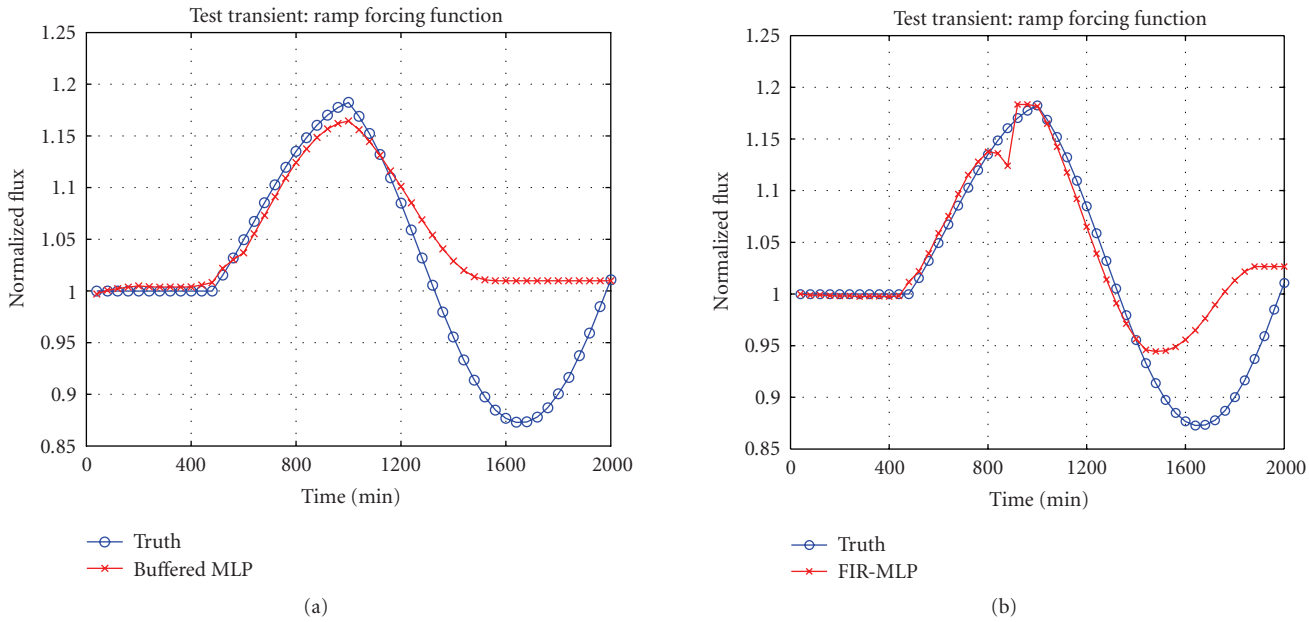



Figure 8: Comparison of the model-simulated normalized flux (circles) with the one estimated by (a) the buffered MLP (crosses) and by (b) the FIR-MLP (crosses) for one sample ramp transient of the test set.

of 0.0039. These values are significantly (7-8 times) lower than those provided by both the buffered MLP (0.0751 and 0.0503, resp.) and the FIR-MLP (0.0698 and 0.0435, resp.); these results are pictorially confirmed by a comparison of Figures 2, 3, and 4 (IIR-LRNN) with Figures 8, 9, and 10 (buffered MLP and FIR-MLP), respectively.

Figures 8, 9, and 10 point out a key disadvantage of the buffer and FIR approaches with respect to the locally recurrent one, that is, the limited past history horizon which prevents modelling of arbitrary long-time dependencies. In this view, the IIR-LRNN represents a generalization of the FIR-MLP to the infinite memory case [28].

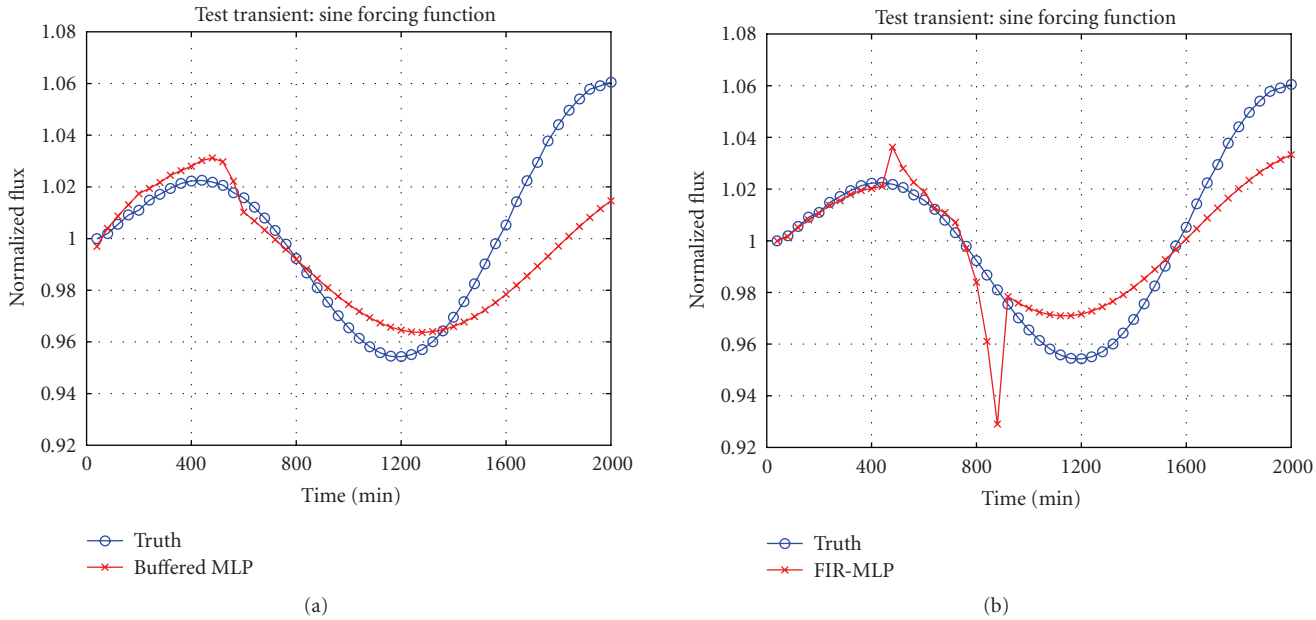



Figure 9: Comparison of the model-simulated normalized flux (circles) with the one estimated by (a) the buffered MLP (crosses) and by (b) the FIR-MLP (crosses) for one sample sinusoidal transient of the test set.

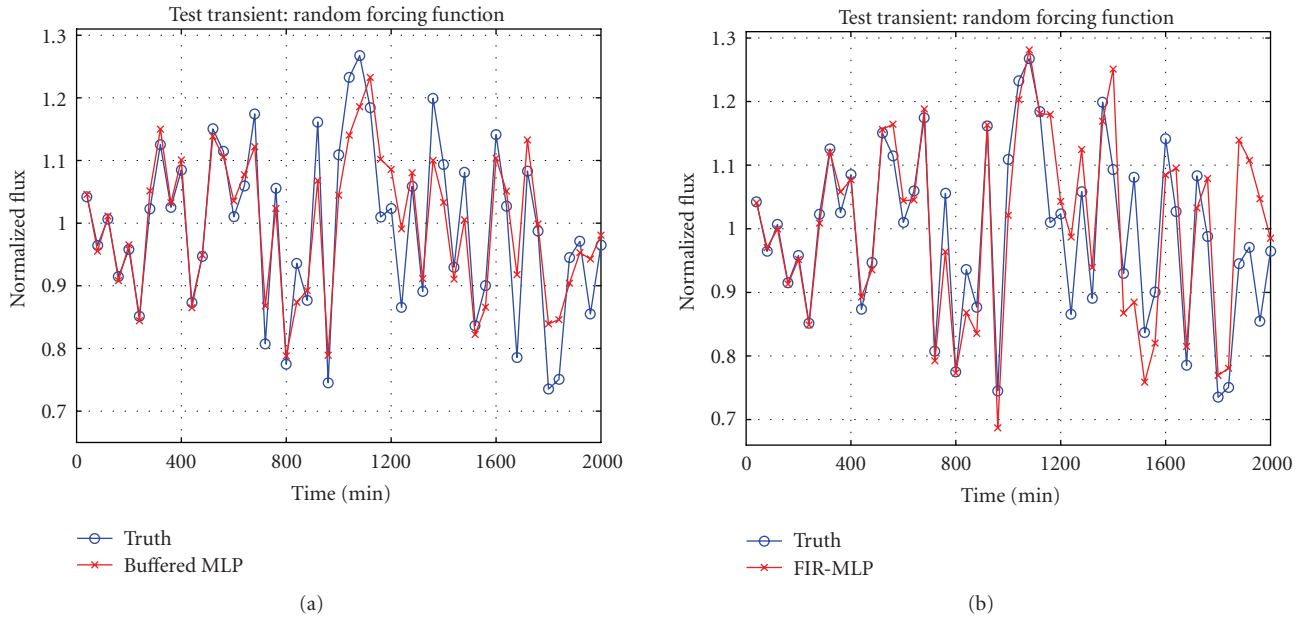



Figure 10: Comparison of the model-simulated normalized flux (circles) with the one estimated by (a) the buffered MLP (crosses) and by (b) the FIR-MLP (crosses) for one sample random transient of the test set.

## 4. CONCLUSIONS

The design, operation, and control of complex industrial systems, such as the nuclear, chemical, and aerospace ones, entail the capability of accurately modelling the nonlinear dynamics of the underlying processes.

In this respect, artificial neural networks (ANNs) have gained popularity as valid alternatives to the lengthy and burdensome analytical approaches for reconstructing complex nonlinear and multivariate dynamic mappings.

In particular, recurrent neural networks (RNNs) are attracting significant attention, because of their intrinsic potentials in temporal processing, for example, time series prediction, system identification and control, and temporal pattern recognition and classification.

In this paper, an infinite impulse response locally recurrent neural network (IIR-LRNN) has been compared to a finite impulse response multilayer perceptron (FIR-MLP) and a conventional static MLP. The comparison has been carried out with respect to the problem of estimating the

evolution of the neutron flux in a simplified nuclear reactor model of literature, starting from the knowledge of reactivity evolution only.

The ability of the trained IIR-LRNN to deal with both the short-term dynamics, governed by the instantaneous variations of the forcing function (i.e., the reactivity), and the long-term dynamics, governed by the *Xe* oscillations, is very satisfactory and turns out to be the main reason for outperforming the static neural modelling approaches of the buffered MLP and FIR-MLP, in terms of both estimation accuracy and generalization capabilities.

## ACRONYMS AND SYMBOLS

ANN: Artificial neural network
RNN: Recurrent neural network
LRNN: Locally recurrent neural network
FIR: Finite impulse response
IIR: Infinite impulse response
MLP: Multilayer perceptron
RBP: Recursive backpropagation
NARX: Nonlinear autoregression with exogenous inputs
AR: Autoregressive
MA: Moving average
RTRL: Real-time recurrent learning
BPTT: Backpropagation through time
RMSE: Root mean square error
MAE: Mean absolute error
MRE: Mean relative error
$k$: Layer index (in particular, $k = 0$ and $k = M$ denote the input and the output layers, resp.)
$N^k$: Number of neurons in the $k$th layer (in particular, $N^0$ and $N^M$ denote the numbers of input and output neurons, resp.)
$J$: Neuron index
$t$: Continuous time index
$x_j^k(t)$: Output of the $j$th neuron of the $k$th layer at time $t$ (in particular, $j = 0$ refers to the bias inputs; note that $x_j^0(t)$, $j = 1, 2, \ldots, N^0$, are the input signals)
$L_{jl}^k - 1$: Order of the MA part of the synapse of the $j$th neuron of the $k$th layer relative to the $l$th output of the $(k-1)$th layer ( $L_{jl}^k \geq 1$ and $L_{j0}^k = 1$)
$I_{jl}^k$: Order of the AR part of the synapse of the $j$th neuron of the $k$th layer relative to the $l$th output of the $(k-1)$th layer ($I_{jl}^k \geq 1$ and $I_{j0}^k = 1$)
$w_{jl(p)}^k$: ($p = 0, 1, \ldots, L_{jl}^k - 1$) coefficients of the MA part of the corresponding synapse (if $L_{jl}^k = 1$, the synapse has no MA part, the weight notation becomes $w_{jl}^k$, and $w_{j0}^k$ is the bias)
$v_{jl(p)}^k$: ($p = 1, 2, \ldots, I_{jl}^k$) coefficients of the AR part of the synapse (if $I_{jl}^k = 0$, the synaptic filter is purely MA)
$f^k(\cdot)$: Nonlinear activation function relative to the $k$th layer
$f_k'(\cdot)$: Derivative of $f^k(\cdot)$
$y_{jl}^k(t)$: Synaptic filter output at time $t$ relative to the synapse connecting the $j$th neuron of the $k$th layer to the $l$th input
$s_j^k(t)$: "Net" input to the activation function of the $j$th neuron of the $k$th layer at time $t$
$d_r(t)$: ($r = 1, 2, \ldots, N^M$) desired target of output node $r$ at time $t$
$\mu$: Learning coefficient
$\alpha$: Momentum coefficient
$n_{\text{epoch}}$: Number of learning epochs during training
$n_{\text{rep}}$: Number of consecutive repetitions of each transient during training
$y(t)$: Generic system output vector at time $t$
$x(t)$: Generic forcing functions vector at time $t$
$\Theta$: Generic set of adjustable parameters of a model
$F(\cdot)$: Mapping function of a process (possibly nonlinear)
$\Phi(t)$: Normalized neutron flux at time $t$
$Xe(t)$: Normalized xenon concentration at time $t$
$I(t)$: Normalized iodine concentration at time $t$
$\rho(t)$: Reactivity value at time $t$
$\Phi_0$: Nominal normalized neutron flux
$Xe_0$: Nominal xenon concentration
$I_0$: Nominal iodine concentration
$\rho_0$: Nominal reactivity value
$\Delta\rho$: Reactivity amplitude variation
$\Sigma_f$: Effective fission macroscopic cross-section ($\text{cm}^{-1}$)
$\sigma_{Xe}$: Effective xenon microscopic cross-section ($\text{cm}^2$)
$\gamma_{Xe}$: Xenon fission yield
$\gamma_I$: Iodine fission yield
$\lambda_{Xe}$: Xenon decay rate ($\text{s}^{-1}$)
$\lambda_I$: Iodine decay rate ($\text{s}^{-1}$)
$\gamma$: Lumped temperature feedback coefficient ($\text{cm}^2\text{s}$)
$C$: Lumped dimensional conversion factor of xenon concentration to reactivity
$\Lambda$: Effective neutron mean generation time (s)
$\hat{\Phi}(t)$: Neural estimate of the normalized neutron flux at time $t$
$N_t$: Number of transients in the training/validation/test sets
$T$: Temporal length of a transient
$\Delta t$: Time step for the numerical simulation of a transient
$n_p$: Number of patterns in a training/validation/test transient ($n_p = T/\Delta t$)
$n$: Transient index
$o$: Pattern index
$T_s$: Steady-state time interval (in step and ramp forcing functions)
$T_v$: Variation time interval (in ramp forcing functions)
$F$: Oscillation frequency (in sinusoidal forcing functions)
$\Phi_{n,o}$: Normalized flux value in the $o$th pattern of the $n$th transient
$\hat{\Phi}_{n,o}L$: Neural estimate of the normalized flux value in the $o$th pattern of the $n$th transient

## REFERENCES


[1] M. Marseguerra and E. Zio, "Monte Carlo approach to PSA for dynamic process systems," *Reliability Engineering & System Safety*, vol. 52, no. 3, pp. 227–241, 1996.

[2] R. Myung-sub, C. Se-woo, and C. Soon-heung, "Thermal power prediction of nuclear power plant using neural network and parity space model," *IEEE Transactions on Nuclear Science*, vol. 38, no. 2, pp. 866–872, 1991.

[3] R. Myung-sub, C. Se-woo, and C. Soon-heung, "Power prediction in nuclear power plants using a back-propagation learning neural network," *Nuclear Technology*, vol. 94, no. 2, pp. 270–278, 1991.

[4] A. G. Parlos, A. F. Atiya, and K. T. Chong, "Nonlinear identification of process dynamics using neural networks," *Nuclear Technology*, vol. 97, no. 1, pp. 79–96, 1992.

[5] D. R. Seidl and R. D. Lorenz, "A structure by which a recurrent neural network can approximate a nonlinear dynamic system," in *Proceedings of International Joint Conference on Neural Networks (IJCNN '91)*, vol. 2, pp. 709–714, Seattle, Wash, USA, July 1991.

[6] H. T. Siegelmann and E. D. Sontag, "On the computational power of neural nets," *Journal of Computer and System Sciences*, vol. 50, no. 1, pp. 132–150, 1995.

[7] K.-I. Funahashi and Y. Nakamura, "Approximation of dynamical systems by continuous time recurrent neural networks," *Neural Networks*, vol. 6, no. 6, pp. 801–806, 1993.

[8] S. Haykin, *Neural Networks: A Comprehensive Foundation*, IEEE Press, New York, NY, USA, 1994.

[9] T. Adali, B. Bakal, M. K. Sonmez, R. Fakory, and C. O. Tsaoi, "Modelling core neutronics by recurrent neural networks," in *Proceedings of World Congress on Neural Networks (WCNN '95)*, vol. 2, pp. 504–508, Washington, DC, USA, July 1995.

[10] D. Back, E. A. Wan, S. Lawrence, and A. C. Tsoi, "A unifying view of some training algorithms for multilayer perceptrons with FIR filter synapses," in *Proceedings of the 4th IEEE Workshop on Neural Networks for Signal Processing (NNSP '94)*, pp. 146–154, Ermioni, Greece, September 1994.

[11] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[12] B. A. Pearlmutter, "Gradient calculations for dynamic recurrent neural networks: a survey," *IEEE Transactions on Neural Networks*, vol. 6, no. 5, pp. 1212–1228, 1995.

[13] A. G. Parlos, K. T. Chong, and A. F. Atiya, "Application of the recurrent multi-layer perceptron in modelling complex process dynamics," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 255–266, 1994.

[14] G. V. Puskorius and L. A. Feldkamp, "Neurocontrol of nonlinear dynamical systems with kalman filter trained recurrent networks," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 279–297, 1994.

[15] M. Boroushaki, M. B. Ghofrani, and C. Lucas, "Identification of a nuclear reactor core (VVER) using recurrent neural networks," *Annals of Nuclear Energy*, vol. 29, no. 10, pp. 1225–1240, 2002.

[16] M. Boroushaki, M. B. Ghofrani, C. Lucas, and M. J. Yazdanpanah, "Identification and control of a nuclear reactor core (VVER) using recurrent neural networks and fuzzy systems," *IEEE Transactions on Nuclear Science*, vol. 50, no. 1, pp. 159–174, 2003.

[17] M. Boroushaki, M. B. Ghofrani, C. Lucas, and M. J. Yazdanpanah, "An intelligent nuclear reactor core controller for load following operations, using recurrent neural networks and fuzzy systems," *Annals of Nuclear Energy*, vol. 30, no. 1, pp. 63–80, 2003.

[18] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Transactions on Neural Networks*, vol. 1, no. 1, pp. 4–27, 1990.

[19] P. Campolucci, A. Uncini, F. Piazza, and B. D. Rao, "Online learning algorithms of locally recurrent neural networks," *IEEE Transactions on Neural Networks*, vol. 10, no. 2, pp. 253–271, 1999.

[20] A. C. Tsoi and A. D. Back, "Locally recurrent globally feedforward networks: a critical review of architectures," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 229–239, 1994.

[21] F. Cadini, E. Zio, and N. Pedroni, "Simulating the dynamics of the neutron flux in a nuclear reactor by locally recurrent neural networks," *Annals of Nuclear Energy*, vol. 34, no. 6, pp. 483–495, 2007.

[22] J. Chernick, "The dynamics of a xenon-controlled reactor," *Nuclear Science and Engineering*, vol. 8, pp. 233–243, 1960.

[23] R. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Computation*, vol. 1, pp. 270–280, 1989.

[24] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, D. E. Rumelhart and J. L. McClelland, Eds., MIT Press, Cambridge, Mass, USA, 1986.

[25] D. Back and A. C. Tsoi, "FIR and IIR synapses, a new neural network architecture for time series modelling," *Neural Computation*, vol. 3, no. 3, pp. 375–350, 1991.

[26] N. Benvenuto, F. Piazza, and A. Uncini, "Comparison of four learning algorithms for multilayer perceptron with FIR synapses," in *Proceedings of IEEE International Conference on Neural Networks (ICNN '94)*, vol. 1, pp. 309–314, Orlando, Fla, USA, June-July 1994.

[27] E. A. Wan, "Temporal backprogagation for FIR neural networks," in *Proceedings of International Joint Conference on Neural Networks (IJCNN '90)*, vol. 1, pp. 575–580, San Diego, Calif, USA, June 1990.

[28] P. Frasconi, G. Gori, and M. Soda, "Local feedback multilayered networks," *Neural Computation*, vol. 4, pp. 120–130, 1992.