Contributed article

Multi-step-ahead prediction using dynamic recurrent neural networks

A.G. Parlosa,*, O.T. Raisa,1, A.F. Atiyab

a Department of Mechanical Engineering, Texas A&M University, College Station, TX 77843, USA
b Learning Systems Group, Department of Electrical Engineering, California Institute of Technology, MS 136-93, Pasadena, CA 91125, USA

Received 31 March 1999; revised 15 June 2000; accepted 15 June 2000

Abstract

A method for the development of empirical predictive models for complex processes is presented. The models are capable of performing accurate multi-step-ahead (MS) predictions, while maintaining acceptable single-step-ahead (SS) prediction accuracy. Such predictors find applications in model predictive controllers and in fault diagnosis systems. The proposed method makes use of dynamic recurrent neural networks in the form of a nonlinear infinite impulse response (IIR) filter. A learning algorithm is presented, which is based on a dynamic gradient descent approach.

The effectiveness of the method for accurate MS prediction is tested on an artificial problem and on a complex, open-loop unstable process. Comparative results are presented with polynomial Nonlinear AutoRegressive with eXogeneous (NARX) predictors, and with recurrent networks trained using teacher forcing. Validation studies indicate that excellent generalization is obtained for the range of operational dynamics studied. The research demonstrates that the proposed network architecture and the associated learning algorithm are quite effective in modeling the dynamics of complex processes and performing accurate MS predictions. © 2000 Elsevier Science Ltd. All rights reserved.

Keywords: Multi-step-ahead prediction; Nonlinear predictors; IIR networks; Dynamic recurrent networks; Dynamic learning; Complex dynamics

1. Introduction

System identification deals with the problem of completing parts of or building an entire model of a system using actual input and output observations. Contrary to time-series prediction, identified models typically include one or more independent inputs, which must be considered in the modeling process. Identified models often find applications in predictive applications, which usually require only single-step-ahead (SS) predictions. In such applications, the predictor uses all or some of the available input and output observations to estimate a variable of interest for the time-step immediately following the latest observation. This, in essence, forms the concept of “closed-loop” prediction, which is limited to estimates of a variable that follow the latest measurements by one time-step. There is a rich literature on methods and algorithms for designing SS predictors, especially when the assumed model structure is linear (Ljung, 1999; Söderström & Stoica, 1998).

In many engineering problems, such as in process controllers utilizing model predictive algorithms, it is required that at any given time the process outputs be predicted many time-steps into the future without the availability of output measurements in the horizon of interest. Additionally, in forecasting and more recently in fault monitoring and diagnosis applications, the availability of accurate empirical models with multi-step-ahead (MS) predictive capabilities are desirable (Gertler, 1998; Natke & Cempel, 1997). MS predictors are difficult to develop because the lack of measurements in the prediction horizon necessitates the recursive use of SS predictors for reaching the end-point in the horizon. Even small SS prediction errors at the beginning of the horizon accumulate and propagate, often resulting in poor prediction accuracy. The situation is even worse for complex systems characterized by poorly understood, noisy and often nonlinear dynamics. Furthermore, the presence of one or more independent process inputs increases the dimensionality of the input space, resulting in a challenging MS prediction problem. Use of linear model structures for MS prediction has not proven very successful in real-world applications.

Over the last ten years, predictor design based on nonlinear model structures has received much attention. Among others, generalizations of the traditional ARX and ARMAX models...
to the nonlinear domain have been proposed, and the polynomial NARX has proven successful in many applications (Billings, Chen, & Korenberg, 1989). More recently, neural networks have been proposed and extensively used in the identification and control of dynamic systems (Chen, Billings, & Grant, 1990; Juditsky, Zhang, Delyon, Glorennec, & Benveniste, 1994; Juditsky et al., 1995; Ljung & Sjoberg, 1992; Narendra & Parthasarathy, 1990; Sjoberg et al., 1995). Most of the published literature, however, considers the application of neural networks for SS prediction.

One of the earliest attempts in using neural networks for long range (or multi-step) prediction was reported by Su, McAvoy and Werbos (1992). This method is based on the backpropagation-through-time (BTT) concept (Werbos, 1990). The BTT approach has more recently been applied to a diverse array of problems, including filtering and classification (Feldkamp & Puskorius, 1998). More recently, Schenker and Agarwal proposed a method for training two distinct networks in order to perform long-range prediction (Schenker & Agarwal, 1995). In their study feedforward networks were used in a feedback configuration. A nonlinear soft-sensor based on MS prediction has been proposed by Yang and Chai (1997), which utilizes a neural network in a nonrecursive form. Cloarec and Ringwood (1998) have proposed the incorporation of statistical forecast averaging methods in MS neural predictive models for time-series. (Prasad, Swidenbank, & Hogg, 1998) have proposed a multivariable long-range predictive control strategy based on a neural network and its application for power plant control. Finally, Atiya, El-Shoura, Shaheen, and El-Sherif (1999) have recently presented the comparison of various neural methods for MS prediction in time-series. We would like to cite also the work by Suykens, De Moor, and Vandewalle (1995). This work does not relate to MS predictors, but it introduces the concept of neural state-space models, a model structure relevant to the dynamic recurrent network used in this study.

Following the early success of feedforward networks in approximating complex static mappings, researchers attempted use of the so-called recurrent networks for the same purpose. Recurrent networks are those that have at least one feedback loop in their structure (Haykin, 1999). Even though the Hopfield network was one of the first recurrent networks used for classification purposes, the recurrent multilayer perceptron (RMLP) network was one of the first networks used in mapping applications. A class of mapping problems that is of great interest in signal processing and control applications involve temporally correlated samples. Such mapping problems are best addressed using dynamic neural networks, that is networks in which either the input and/or the output network layer contains temporally correlated samples (Haykin, 1999). For example, both a finite-duration impulse response (FIR) network and an infinite-duration impulse response (IIR) network are considered dynamic. Either feedforward or recurrent networks can be combined with a dynamic architecture to obtain dynamic (feedforward) or dynamic recurrent neural networks, respectively. Earlier attempts by the authors to use recurrent networks for MS prediction resulted in predictors that were effective in some multi-step transients, though they were very sensitive to process and sensor noise. These predictors performed somewhat poorly during certain transients (Parlos, Chong, & Atiya, 1994). To improve the MS prediction performance, the network parameters had to be updated for specific transients using on-line learning, indicative of the insufficient predictive performance of the off-line learning phase. The predictor developed in that study was based on recurrent neural networks with only “local feedback” (LF). No delayed values of the measured system input or output were utilized in the network input layer. In view of the previous definitions, the predictor was static, though the presence of local feedback in the recurrent hidden layer nodes introduced predictor dynamics. In system theoretic terms, such a predictor is in the form of an empirical state-space in the so-called noise representation (Ljung, 1999).

In this paper we propose MS predictors that utilize a dynamic recurrent network in the form of a (nonlinear) IIR filter. Current and delayed (or past) observations of the measured system input and output are utilized as inputs to the network. We say that this architecture includes “global feedback” (GF), in addition to the LF of the hidden layer nodes. Training of this network is equivalent to the parallel identification scheme discussed by Narendra and Parthasarathy (1990). Training is performed by adapting a learning algorithm previously developed by the authors Parlos et al. (1994), for use in MS prediction. The advantages that such a learning algorithm offers, compared say to BTT, is the ease by which on-line training can be performed. In this paper, the effectiveness of the MS prediction is demonstrated via simulations on a simple artificial problem, followed by extensive simulations on an open-loop unstable process system with highly complex dynamics. The lack of any rigorous proofs regarding the superiority of MS predictors over SS predictors necessitates the use of real-world case studies to convincingly demonstrate the relevant points. This is especially true in the case of nonlinear model structures with independent inputs, as contrasted to time-series models. Furthermore, in this study a comparison is made with a nonneural network predictor, namely a polynomial NARX, to demonstrate the superiority of MS over SS predictors. The lack of polynomial-type MS predictors have prevented us from presenting comparisons with a nonneural MS predictor.

The contributions of this paper are as follows.

- The MS prediction problem is addressed using dynamic recurrent neural networks, and comparisons are made with SS predictions. In this process, a dynamic learning algorithm previously developed by the authors: Parlos et al. (1994) is adapted for use in MS prediction, accounting for the impact of GF. This is in addition to the LF present in recurrent networks.
The effectiveness of the developed learning algorithm in training accurate MS predictors is demonstrated by modeling a complex process system, characterized by poorly understood two-phase fluid flow effects, as well as reverse and unstable open-loop dynamics. The performance of these predictors is extensively tested. Comparative simulations with another empirical model are presented demonstrating the effectiveness of the proposed approach. It should be noted that because of the open-loop process instability, the test results presented demonstrate the feasibility of operating the neural predictor and the process controller in a feedback loop configuration.

As mentioned in the foregoing discussion, accurate MS predictors find many important applications. However, the results of this paper demonstrate that predictors designed by minimizing a MS prediction error criterion result in models with better generalization capabilities, whether intended for use in SS or MS applications.

The remainder of this paper is organized as follows. Section 2 presents the nonlinear MS prediction problem considered, and its solution using a dynamic recurrent network. Section 3 presents the learning algorithm used in training the dynamic neural network. Section 4 presents the MS prediction results for an artificial problem. The results obtained from using the polynomial NARX and the study. In fact, in the context of an ARMA process, and under the assumption of Gaussian noise, it can be shown that the optimal mean-square MS prediction can be obtained by a recursive procedure such as this (Söderström & Stoica, 1998). Accurate MS prediction capability implies accurate modeling of the deterministic dynamics of a system. In fact, performing MS prediction (also called simulation) tests for an identified model constitutes one of the many available model validation tests (Ljung, 1999).

The recursive relation between inputs and outputs in MS prediction can be expressed using general nonlinear input–output models, as follows:

\[
\hat{y}(k + p; k) = f(y(k + p - 1); k), \\
\hat{y}(k + p - 2; k), ..., y(k), ..., y(k - n), \\
\hat{u}(k + p - 1; k), \hat{u}(k + p - 2; k), ..., u(k), ..., u(k - m)),
\]

where \( p \) is the MS prediction horizon, \( n \) and \( m \) are the number of delayed outputs and inputs used in the model, respectively. The functional form \( f(\cdot) \) in Eq. (1) can be approximated using a feedforward network. The resulting model belongs to the class of dynamic networks, because of the presence of IIR-type GF. Out attempts to use a feedforward network with a dynamic learning algorithm for accurate MS prediction did not succeed in its entirety. Even though accurate MS predictions were obtained for relatively simple problems, for the complex process system presented in this study the results were not acceptable no matter what combination of network architecture, delayed inputs and outputs were selected. The prediction accuracy deteriorated very quickly with increased \( p \).

An alternate approach to obtaining an MS prediction model is to use recurrent (rather than a feedforward) networks. This corresponds to obtaining a “state-space” model of the identified process, where the quotes denote that we will not be dealing with a physical state-space, rather it will be empirical. Here, we define a state-space as physical if the state variables used in the model represent actual physical quantities, such as temperature, pressure, etc. Such models usually result from physics-based models used in engineering. An empirical state-space is one in which the state variables do not have any physical significance, much like the coefficients of an empirical input–output model. In empirical modeling our interest is in preserving the significance and accuracy of the model inputs and outputs.

For a general discrete-time stochastic nonlinear system, it is widely known in the literature that SS predictors can be expressed using so-called innovations representation. In particular, the directly parametrized innovations

---

2 Unless otherwise stated, all developments are presented in the discrete-time domain.
representation can be expressed as follows (Ljung, 1999; Suykens et al., 1995):

\[
\begin{align*}
\dot{x}(k + 1/k) &= f(\hat{x}(k/k), u(k); \theta) + K(\theta)e(k), \\
\dot{y}(k + 1/k) &= g(\hat{x}(k + 1/k); \theta),
\end{align*}
\]  

(2)

where, the quantities with a "hat" represent estimates of the actual states and outputs, and where \(e(k)\) is the innovations term defined as the difference between the observed and predicted value of the system output, i.e., \(e(k) = y(k) - \hat{y}(k/k - 1)\). An alternate form for this predictor can be obtained using the Extended Kalman Filter (EKF) (Ljung, 1999). The effective computation of the parameters of such predictors for real-world problems remains a challenge.

Suykens proceeds by approximating the functionals in Eq. (2) using feedforward networks (Suykens et al., 1995) calling the resulting form a neural state-space representation.

It is worth observing that the predictor form of Eq. (2) includes the most recent value of the innovations term, \(e(k)\). The underlying assumption here is that the dimension of the predictor state (or the predictor order) is equal to the dimension of the system state being identified. In real-world applications this is far from being the case. The dimension of the predictor state will usually be much smaller than the dimension of the actual system being identified, which might be infinite. In such circumstances, utilizing only the most recent value of the innovations term in the predictor is not sufficient. Rather, additional past values of the innovations term (or more generally, the observations and predictions) must be utilized, as also stated by Ljung (1999).

In this study we are primarily concerned with exploring systems that have complex dynamics characterized by possibly an infinite dimensional state-space. As a result, no accurate assumptions can be made regarding the exact value of the dimension of the physical state-space that will lead to accurate MS predictions. Furthermore, we are not interested in estimating the physical states of the systems. Rather, we are concerned about the MS prediction of the system outputs. Since the exact number of system states needed for accurate MS prediction will not be known, in an empirical state-space set-up we are forced to use many delayed input and output observations, just as in the case of the predictor given by Eq. (1). We propose using the following \(p\)-step-ahead predictor form:

\[
\begin{align*}
\dot{z}(k + p/k) &= \mathcal{F}(z(k + p - 1/k), \hat{U}(k + p - 1/k); W), \\
\dot{y}(k + p/k) &= \mathcal{G}(z(k + p/k); W),
\end{align*}
\]

(3)

where \(z(k + p/k)\) is the conditional value of the empirical (or artificial) state vector, given observations up to time \(k\). The state vector, \(z(\cdot)\), has no relation whatsoever to the previously defined physical state vector, \(x(\cdot)\). The parameters in the functionals \(\mathcal{F}(\cdot)\) and \(\mathcal{G}(\cdot)\) are to be determined by an estimation process, following some structural assumptions. The vector \(\hat{U}(k + p - 1/k)\) is defined as

\[
\hat{U}(k + p - 1/k) = [\hat{y}(k + p - 1/k), \ldots, \hat{y}(k + 1/k), \ldots, \hat{y}(k - m)]^T,
\]

(4)

where the \(\hat{u}(\cdot)\)s and \(\hat{y}(\cdot)\)s are estimates of the observed physical system inputs and outputs, and where the matrix \(W\) includes all of the predictor parameters to be estimated. The system inputs during the prediction horizon are best estimates of their anticipated values. If \(p > n\) or \(p > m\) then the vector \(\hat{U}(\cdot)\) is defined in Eq. (4). Otherwise, the estimates contained in \(\hat{U}(\cdot)\) that are at times prior to the beginning of the prediction horizon, are replaced with actual observations. In the next subsection, we demonstrate that the dynamic recurrent network used in this study can be cast in the form of an MS predictor of Eq. (3).

Having selected the model structure to be used in MS prediction, we can now proceed to formulate the relevant objective function. In this formulation, we allow for the prediction horizon to be a possibly moving window of \(W\) points. For a system with \(K\) outputs to be predicted, the following objective function is optimized:

\[
\begin{align*}
\delta(t) &= \sum_{k=t+1}^{t+W} E(t, k) = \sum_{k=t+1}^{t+W} \sum_{j=1}^{K} (\hat{y}_j(k) - y_j(k))^2,
\end{align*}
\]

(5)

where the error \(E(t, k)\) depends both on the window location and the prediction point within the window. For off-line learning we select \(t = 1\) and \(W = NP\), where \(NP\) is the size of the training set, usually much larger than the largest \(p\) needed in practice. For on-line learning \(W\) can be selected to be sufficiently large so as to include the largest possible prediction horizon.

### 2.2. Dynamic recurrent neural network architecture

The network used in this study to optimize the objective function (5) is the Recurrent Multilayer Perceptron (RMLP) network and it is based on prior work by the authors: Fernandez, Parlos, and Tsai (1990) and Parlos et al. (1994).

The equations describing the \(i^{th}\) node in the \(l^{th}\) layer of the network are as follows:

\[
\begin{align*}
\dot{z}_{l,[i]}(k + 1) &= \sum_{j=1}^{N(l)} w_{l,[i],j}x_{l,j}(k) + \sum_{j=1}^{N(l-1)} w_{l-1,[i],j}x_{l-1,j}(k + 1)b_{l,[i]}, \\
x_{l,[i]}(k) &= f_{l,[i]}(z_{l,[i]}(k)),
\end{align*}
\]

(6)

where \(z_{l,[i]}(k)\) and \(x_{l,[i]}(k)\) represent the state and output variables of the \(i^{th}\) node in the \(l^{th}\) layer, respectively; \(b_{l,[i]}\) is the bias to the node; \(w_{l,[i],j}\) is the weight associated with the link between the \(j^{th}\) node of the \(l^{th}\) layer to the \(i^{th}\) node of the \(l'\) layer, with \(i = 1, \ldots, N(l)\) (the number of nodes per layer) and \(l = 1, \ldots, L'\) (number of layers). The nonlinear
network parameters are defined as the ahead prediction of the observed system output. The network input vector, \( \hat{x}(k) \), and the network output vector, \( \hat{y}(k) \), are the discriminatory functions of the hidden layer, with \( \text{discriminatory functions of the hidden layer, with } z = 1, \ldots, N(3) \), the network output vector, \( \hat{y}(k) \), and the network output vector, \( \hat{y}(k) \), being the number of hidden nodes. The conditional value of the hidden layer state vector given observations up to time \( k \) is also an \( N(2) \) dimensional vector and it is defined by \( z(k + p/k) = [z_{1}(k + p/k), \ldots, z_{N(2)}(k + p/k)]^{T} \). The network input vector, \( \hat{U}(k + p - 1/k) \), is defined by Eq. (4) and the network output vector, \( \hat{y}(k + p/k) \), is the \( p \)-step-ahead prediction of the observed system output. The network parameters are defined as \( \theta_{i,j} = [b_{i,j,1}, \ldots, b_{i,j,N(j)}]^{T} \), for \( j = 2 \) and 3, where \( \theta_{2,3} = [w_{i,2,N(3)}; i = 1, \ldots, N(1); j = 1, \ldots, N(2)] \) and \( \theta_{2,2} = [w_{i,2,N(2)}; i = 1, \ldots, N(1); j = 1, \ldots, N(2)] \) are the feedforward weights, and where \( \theta_{2,3} = [w_{i,2,N(2)}; i = 1, \ldots, N(1); j = 1, \ldots, N(2)] \) are the local feedback (or recurrent) weights. The network input dimension is \( N(1) = m \times q + n \times s \), and the network output dimension is \( N(3) = s \). Eq. (8) is structurally similar to the predictor form of Eq. (3), where the nonlinear functional \( \mathcal{F}(\cdot) \) is set to \( \mathcal{F}(\cdot) \) and where \( \mathcal{G}(\cdot) \) is a linear function of the state \( z(k + p/k) \). The predictor parameter matrix \( \mathbf{W} \) consists of \( \mathcal{W}_{1,2}, \mathcal{W}_{2,2}, \mathcal{W}_{2,3}, b_{b_{1}}, \) and \( b_{b_{3}} \). The dynamic recurrent neural network, described by Eqs. (8)–(13), is depicted in Fig. 1.

We now turn our attention to the learning (or parameter estimation) algorithm that will minimize the objective function (5), subject to the constraints of the predictor structure given by Eq. (8).

### 3. Multi-step-ahead prediction learning algorithm

Recurrent network learning algorithms can be grouped into five major categories, as follows: (1) the forward propagation method, also known as the real time recurrent learning; (2) the BTT method; (3) the fast forward propagation method; (4) the Green’s function method; (5) the block update method. A more detailed and structured overview of recurrent network learning algorithms can be found in Atiya and Parlos (in press). The algorithm derived in this section is based on the forward propagation method developed by Williams and Zipser (1989).

In performing \( p \)-step-ahead prediction recursively, it has been our experience that a dynamic recurrent network has been more effective than a dynamic feedforward network. In
order to derive the error gradients for the dynamic recurrent network without having to unfold the network as it is done in BTT, it is necessary to introduce an additional assumption. In the set-up used in this study, a number of predictor inputs are dependent upon each other. For example, $\hat{y}(k/t)$ is dependent upon $\hat{y}(k - 1/t)$ and so on. This complication can be circumvented in two ways; either by assuming that $\hat{y}(k/t), ..., \hat{y}(k - n_t + 1/t)$ are outputs of other dynamic recurrent predictors or by assuming that

$$
\begin{align*}
\hat{y}(k - 1/t) &= y(k - 1), \\
\hat{y}(k - 2/t) &= y(k - 2), \\
&\vdots \\
\hat{y}(k - n/t) &= y(k - n).
\end{align*}
$$

(14)

That is all past predictions, with the exception of the latest prediction $\hat{y}(k/t)$, are replaced by their equivalent targets from the training set. Thus, the predictor is composed of two types of input nodes; those that consist of all independent inputs, i.e. $y(k - 1), ..., y(k - n), u(k), ..., u(k - m)$, and the prediction node $\hat{y}(k/t)$. Note that even with the aforementioned assumption, the $p$-step-ahead prediction error is being computed for both the learning and recall modes of the network. Furthermore, during network recall the network predictions are used instead of their equivalent targets.

The training set can be expressed as follows:

$$
S = \{ (u_i(k), y_j(k)), \forall k = 1, ..., NP; \, i = 1, ..., N(1); \\
\}
j = 1, ..., N(\mathcal{I}).
$$

(15)

It is desired to minimize the objective function expressed by Eq. (5). The learning algorithm is derived for a general recurrent multilayer network with multiple hidden layers. For the remainder of the derivation the variable $t$ is omitted from the equations for convenience. It should be reiterated that the independent variable $t$ represents the trailing edge of the moving time-window over which the MS prediction error is minimized.

The network weights and biases are updated using the following gradient descent rules:

$$
\Delta W_{[l-1,j][l,i]} = -\eta \sum_{k=t+1}^{t+W_r} \left( \frac{\partial E(k)}{\partial W_{[l-1,j][l,i]}} \right),
$$

(16)

$$
\Delta W_{[l,j][l,i]} = -\eta \sum_{k=t+1}^{t+W_r} \left( \frac{\partial E(k)}{\partial W_{[l,j][l,i]}} \right),
$$

(17)

$$
\Delta b_{[l,j]} = -\eta \sum_{k=t+1}^{t+W_r} \left( \frac{\partial E(k)}{\partial b_{[l,j]}} \right),
$$

(18)

where $\eta$ is the learning rate. For the on-line learning mode, there is no predetermined training set, and the weight updating must be performed as sensed information is received.

The equations describing the $i$th node of the $l$th layer of the network are given by Eq. (6). In particular, the equations describing the $i$th node of the first hidden layer can be written as

$$
\begin{align*}
z_{[2,j]}(k) &= \sum_{j=1}^{N(\mathcal{I})} w_{[2,j][2,j]} x_{[2,j]}(k - 1) \\
&\quad + \sum_{j=1}^{N(\mathcal{I})} w_{[1,j][2,j]} x_{[1,j]}(k) \\
&\quad + \sum_{j=N(\mathcal{I})+1}^{N(\mathcal{I})+N(\mathcal{I})} w_{[1,j][2,j]} x_{[1,j]}(k - 1/t) + b_{[2,j]}, \\
x_{[2,j]}(k) &= F_{[2]}(z_{[2,j]}(k)).
\end{align*}
$$

(19)

Note that comparing Eq. (19) with Eq. (6), an additional third term is present which accounts for the latest prediction utilized in the input layer.

In view of the prediction error defined by Eq. (5), the error gradient with respect to the weights and the biases can be obtained by using the chain-rule, as follows:

$$
\frac{\partial E(k)}{\partial W_{[l,j][l,i]}} = 2 \sum_{n=1}^{N(\mathcal{I})} \left( \hat{y}_n(k/t) - y_n(k) \right) \frac{\partial \hat{y}_n(k/t)}{\partial W_{[l,j][l,i]}},
$$

(20)

$$
\frac{\partial E(k)}{\partial W_{[l-1,j][l,i]}} = 2 \sum_{n=1}^{N(\mathcal{I})} \hat{y}_n(k/t) - y_n(k) \frac{\partial \hat{y}_n(k/t)}{\partial W_{[l-1,j][l,i]}},
$$

(21)

$$
\frac{\partial E(k)}{\partial b_{[l,j]}} = 2 \sum_{n=1}^{N(\mathcal{I})} \left( \hat{y}_n(k/t) - y_n(k) \right) \frac{\partial \hat{y}_n(k/t)}{\partial b_{[l,j]}},
$$

(22)

The recursion equations used to execute the forward gradient propagation can be derived by differentiating Eqs. (19) and (6) with respect to $w_{[l,j][l,i]}, w_{[l-1,j][l,i]}$, and $b_{[l,j]}$, respectively. These differentiations result in equations described in the following development. In what follows, let $x_{[1,N(1)-N(\mathcal{I})+m]}(k)$ equal to $x_{[1,N(1)-N(\mathcal{I})+m]}(k)$. Then we have, $(\partial x_{[1,N(1)-N(\mathcal{I})+m]}(k)/\partial w_{[l,j][l,m]}) = 0$, for $i = 1, ..., N(1) - N(\mathcal{I}), j = 1, ..., N(l), l = 2, ..., \mathcal{I}, m = 1, ..., N(l)$, and $l' = l - 1$ or $l' = l$.

For the gradients with respect to the network connections within the layer, namely, the recurrent and cross-talk weights, we have the following equation, valid for all
hidden layers and output layer, namely, \( 1 < l' \leq \mathcal{L} \):

\[
\frac{\partial x_{l'[n]}(k)}{\partial w_{l,ij}(l)} = \begin{cases} 
F'_{l'}(z_{l'[n]}(k)) \left[ \sum_{m=1}^{|\mathcal{N}(l')} w_{l'[m],n}(l') \frac{\partial x_{l'[m]}(k-1)}{\partial w_{l,ij}(l)} \right] \\
+ \sum_{m=1}^{|\mathcal{N}(l')-1|} w_{l'[m-1],n}(l') \frac{\partial x_{l'[m-1]}(k)}{\partial w_{l,ij}(l)} \\
+ \delta_{m}x_{l'[n]}(k-1) & \text{if } l' \neq l, \\
F'_{l'}(z_{l'[n]}(k)) \left[ \sum_{m=1}^{|\mathcal{N}(l')} w_{l'[m],n}(l') \frac{\partial x_{l'[m]}(k-1)}{\partial w_{l,ij}(l)} \right] \\
+ \sum_{m=1}^{|\mathcal{N}(l')-1|} w_{l'[m-1],n}(l') \frac{\partial x_{l'[m-1]}(k)}{\partial w_{l,ij}(l)} \\
+ \delta_{m}x_{l'[n]}(k) & \text{if } l' = l. 
\end{cases}
\]  

(23)

Next, we compute the gradient for the interlayer connections, namely \( \frac{\partial x_{l'|n}(k)}{\partial w_{l-[1],ij}} \). For \( 1 < l' \leq \mathcal{L} \), the gradient with respect to interlayer connections is computed as follows:

\[
\frac{\partial x_{l'|n}(k)}{\partial w_{l-[1],ij}} = \begin{cases} 
F'_{l'}(z_{l'|n}(k)) \left[ \sum_{m=1}^{|\mathcal{N}(l'|1)} w_{l'[m],n}(l') \frac{\partial x_{l'[m]}(k-1)}{\partial w_{l-[1],ij}} \right] \\
+ \sum_{m=1}^{|\mathcal{N}(l'|1)-1|} w_{l'[m-1],n}(l') \frac{\partial x_{l'[m-1]}(k)}{\partial w_{l-[1],ij}} \\
+ \delta_{m}x_{l'|n}(k-1) & \text{if } l' \neq l, \\
F'_{l'}(z_{l'|n}(k)) \left[ \sum_{m=1}^{|\mathcal{N}(l'|1)} w_{l'[m],n}(l') \frac{\partial x_{l'[m]}(k-1)}{\partial w_{l-[1],ij}} \right] \\
+ \sum_{m=1}^{|\mathcal{N}(l'|1)-1|} w_{l'[m-1],n}(l') \frac{\partial x_{l'[m-1]}(k)}{\partial w_{l-[1],ij}} \\
+ \delta_{m}x_{l'|n}(k) & \text{if } l' = l. 
\end{cases}
\]  

(24)

The gradients with respect to the biases are given by the following equation:

\[
\frac{\partial x_{l'|n}(k)}{\partial b_{l,j}} = \begin{cases} 
F'_{l'}(z_{l'|n}(k)) \left[ \sum_{m=1}^{|\mathcal{N}(l')} w_{l'[m],n}(l') \frac{\partial x_{l'[m]}(k-1)}{\partial b_{l,j}} \right] \\
+ \sum_{m=1}^{|\mathcal{N}(l')-1|} w_{l'[m-1],n}(l') \frac{\partial x_{l'[m-1]}(k)}{\partial b_{l,j}} \\
+ \delta_{m}x_{l'|n}(k-1) & \text{if } l' \neq l, \\
F'_{l'}(z_{l'|n}(k)) \left[ \sum_{m=1}^{|\mathcal{N}(l')} w_{l'[m],n}(l') \frac{\partial x_{l'[m]}(k-1)}{\partial b_{l,j}} \right] \\
+ \sum_{m=1}^{|\mathcal{N}(l')-1|} w_{l'[m-1],n}(l') \frac{\partial x_{l'[m-1]}(k)}{\partial b_{l,j}} \\
+ \delta_{m}x_{l'|n}(k) & \text{if } l' = l. 
\end{cases}
\]  

(25)

Eqs. (23)–(25) are valid for all layers \( l' \), \( l = 2, ..., \mathcal{L} \), and nodes \( i, j = 1, ..., N(l) \), sweeping the entire network, and where

\[
\delta_{m} = \begin{cases} 
1 & \text{if } i = n, \\
0 & \text{otherwise}. 
\end{cases}
\]  

(26)

Note that \( N(1) \) includes the number of current and delayed inputs of the system as well as the number of delayed output(s) and prediction utilized as input to the network. Furthermore, the initial values for the gradients \( (\partial x_{l'|n}(0)/\partial w_{l-[1],ij}) \), \( (\partial x_{l'|n}(0)/\partial w_{l-[2],ij}) \), \( (\partial x_{l'|n}(0)/\partial w_{l-[1],ij}) \) and \( (\partial x_{l'|n}(0)/\partial b_{l,j}) \) are set to zero. Eqs. (23)–(25) must be applied separately for each weight and bias in the network.

In addition to the learning algorithm equations, there are many issues that must be addressed regarding the network architecture and convergence of the learning algorithm to a set of acceptable weights and biases. Predictors that have good generalization capabilities, as manifested by good MS prediction, are desired. To a large extent, the process of obtaining a network architecture is still ad-hoc, especially for real-world problems. These issues are addressed in the application sections of the study.

### 4. An artificial problem

In this section MS predictors are developed for a relatively simple artificial problem. Three different predictors are developed, which are then tested for SS and MS prediction accuracy. One of the predictors is based on the polynomial NARX approximation developed by Billings and his colleagues (Chen & Billings, 1989a,b). The other two predictors are based on the recurrent network presented in previous sections; one with all observations used in the input layer (denoted as TF, for teacher forcing or recurrent network) and one with past predictions used in the input
layer (denoted as GF, for global feedback or dynamic recurrent network). The comparison of the neural MS predictors with the polynomial NARX predictor is somewhat unfair because the latter is derived by minimizing a SS prediction objective function. However, no simple formulation is available for the polynomial NARX MS predictors for use in a more fair comparison. In this section, only a few selected predictor testing results are presented for this artificial problem. More extensive testing of the predictors has been performed with similar satisfactory and consistently good results (Rais, 1995).

In this paper, the following two relative root mean-squared-errors (RRMSE) are used for SS and MS prediction testing error calculations, respectively:

\[
\text{RRMSE}_1(t) = \frac{\sqrt{\frac{1}{t+N} \sum_{k=t+1}^{t+N} (\hat{y}(k|k-1) - y(k))^2}}{\sqrt{\frac{1}{t+N} \sum_{k=t+1}^{t+N} (y(k) - \bar{y})^2}},
\]

(27)

\[
\text{RRMSE}_2(t) = \frac{\sqrt{\frac{1}{t+N} \sum_{k=t+1}^{t+N} (\hat{y}(k|t) - y(k))^2}}{\sqrt{\frac{1}{t+N} \sum_{k=t+1}^{t+N} (y(k) - \bar{y})^2}},
\]

(28)

where \(N\) is the number of data points considered in the error calculation and where the prediction horizon is assumed to span the interval from \(t + 1\) to \(t + N\). \(\bar{y}\) is the observed output vector, \(\hat{y}\) is the predicted output vector and \(\bar{y}\) is the sample mean of the observed output vector. For the results presented in the next sections, the error Eqs. (27) and (28) are used, with \(t = 0\).

4.1. Problem description and predictor development

The problem studied in this section is expressed by the following state-space equations:

\[
\begin{align*}
x_1(k+1) &= 0.5(x_1^2(k))^{\frac{1}{2}} + 0.3x_2(k)x_3(k) + 0.2u_1(k) + n_{1\text{process}}(k), \\
x_2(k+1) &= 0.5(x_2^2(k))^{\frac{1}{2}} + 0.3x_1(k)x_1(k) + 0.5u_1(k) + n_{2\text{process}}(k), \\
x_3(k+1) &= 0.5(x_3^2(k))^{\frac{1}{2}} + 0.3x_1(k)x_2(k) + 0.5u_2(k) + n_{3\text{process}}(k), \\
y_1(k) &= 0.5(x_1(k) + x_2(k) + x_3(k)), \\
y_2(k) &= 2x_1^2(k),
\end{align*}
\]

where \(n_{i\text{process}}(k)\) for \(i = 1, 2, 3\) is zero-mean white Gaussian process noise, with 0.05 standard deviation, defined as low noise level, and with 0.08 standard deviation, defined as high noise level.

For the neural predictors, the training set consisted of all possible 25 combinations of step inputs with magnitude 0.125, 0.25, 0.375, and 0.5, as well as the zero input, for both input channels. Each signal segment consists of 15 samples, for a total 375 samples. Additionally, five different pulses of suitable shapes, each containing 40 samples, were added to the training set. The training set contained a total of 575 samples. The process noise for the entire training set is set at the low noise level. For neural predictor development, the training set is split into an estimation and a cross-validation set. The algorithm presented in Section 3 is applied to the estimation set, while the cross-validation set is used to terminate training and determine the network architecture, both the number of hidden layer nodes, and number of input and output delays.

The NARX-based predictor is developed using the entire training set summarized above, consisting of the 575 data points. The terms of the polynomial NARX predictor were allowed to be at most cubic with delays up to 100. The reason for allowing such high order delays is our desire not to compute the relative order of the system being modeled. Real-world processes are infinite dimensional systems described by partial differential equations, and assuming that one could compute their relative order is quite unrealistic. The order selection algorithm reported by Chen and Billings (1989b) is used, resulting in the following model expressed in the form of an SS predictor

\[
\begin{align*}
\dot{y}_1(k+1) &= 0.79y_1(k) + 0.3u_1(k) - 0.43y_1(k-2)u_1(k-1) + 1.3u_2^2(k-1)u_2(k) + 0.17y_1(k-2), \\
\dot{y}_2(k+1) &= 0.69y_2(k) + 0.44y_1(k)u_1(k) + 0.04y_1(k-3) + 1.08u_1^2(k-2)u_2(k-2) - 0.66u_1(k-3)u_1(k-1)u_2(k-1).
\end{align*}
\]

(30)

This predictive model is also used for MS predictions.

4.2. Simulated predictor testing results

Simulated predictor testing was performed with signals that are completely different than the ones included in the training set. The first test signal is \(u_1(k) = 0.3 + 0.2\sin(\pi k/8)\), and \(u_2(k) = 0.2\), where the step input in the second channel is delayed by five time steps with respect to the input in the first channel. The second test signal is a step with magnitude 0.3 and 0.2 for the first and second channels, respectively. The third test signal is a pulse of \(u_1(k) = 0.3\)
for $k \leq 25$ and $u_2(k) = 0.3$ for $10 \leq k \leq 35$. Each of the three test signals is augmented with zero mean, white Gaussian noise with standard deviation corresponding to low and high noise levels.

The three test signals are applied to the polynomial NARX predictor of Eq. (30). This predictor is the result of one training attempt accomplished using the training data presented in the previous section. Even though the polynomial NARX predictor has been developed by minimizing a SS prediction error, we perform both SS and MS prediction tests using the same polynomial NARX predictor. There is no simple analytic formulation for the polynomial NARX MS predictor that can be utilized. The 100-step-ahead prediction results for the first and second outputs of this predictor are shown in Figs. 2 and 3, respectively. The polynomial NARX-based SS and MS prediction errors are summarized in Tables 1 and 2, respectively. The errors presented in Tables 1 and 2 have been computed using Eqs. (27) and (28), respectively.

The same test signals are used to test the two neural predictors. The recurrent neural predictor trained using TF minimizes a SS prediction error, while the dynamic recurrent neural predictor trained with GF minimizes a MS prediction error. The MS predictor results using the predictor trained with GF are shown in Figs. 4 and 5. The prediction errors for these simulations are summarized in Tables 1 and 2. In these tables also included are the MS prediction errors for the neural predictor trained using TF.

The results in Tables 1 and 2 indicate that, for the three test signals applied to MS prediction, the neural predictor trained with TF is more accurate than the polynomial NARX and the neural predictor trained with GF. Conversely, for MS predictions the neural predictor trained with GF is more accurate than the other two predictors. As the training set noise is increased the accuracy of all predictors deteriorates. Nevertheless, in this artificial problem the incremental improvement obtained by using the dynamic recurrent neural predictor for MS prediction simulations is relatively small, and it might not justify use of a complex learning algorithm such as the one presented in Section 3.

Fig. 2. Multi-step-ahead response of the first output of the NARX predictor to high noise (top, sinusoidal input; middle, step input; bottom, pulse input).
As the next section reveals, however, with increased process complexity the simulation results support an entirely different conclusion.

5. A complex process system

In this section a complex process is studied, namely a Westinghouse-type U-tube steam generator (UTSG)\(^3\). This heat exchanger is a critical process component widely encountered in nuclear power plants. A schematic of the UTSG, with some of its typical physical dimensions is shown in Fig. 6. A slightly modified version of the error defined by Eq. (27) is used; the sample mean of the observations in the denominator of the error definition is set to zero. Only a few selected predictor testing results are presented for this process system. More extensive testing of the developed predictors has been performed, with similar satisfactory and consistently good results (Rais, 1995).

5.1. Physical process description

High pressure liquid from the power plant primary loop flows in and out of the process system primary side (tube side). The primary side liquid flow path is up from the hot leg inlet, to a semicircular turnaround, and then down to the cold leg exit. Feed-water enters the process system from the steam plant. At this junction, the feed-water mixes with the liquid being discharged from the liquid–vapor separation devices. The liquid mixture flows downward through the annular downcomer region. After a turn at the bottom of the downcomer, the liquid is heated during an upward passage outside the U-tubes in the tube bundle region. The heat transferred across the tube bundles causes evaporation of some of the liquid, and a two-phase mixture exits from the tube bundle entering the riser. The liquid and vapor are partially separated by swirl vanes at the top of the

\(^3\) In the remainder of this section the UTSG is referred to as “the process system”.

Fig. 3. Multi-step-ahead response of the second output of the NARX predictor to high noise (top, sinusoidal input; middle, step input; bottom, pulse input).
riser and more separation occurs near the top of the process system. The vapor that results from this is saturated with a quality of about 99.75%. This vapor is then fed to the turbine units and other auxiliary equipment (Rais, 1995).

An existing process simulator developed by Strohmayer (1982), and modified by Choi (1987) was adopted for the purpose of this study. The process simulator was developed using one dimensional mass, momentum, and energy conservation equations. An integrated secondary-recirculation-loop momentum equation is incorporated into the simulator to calculate the water level. The simulator has been successfully validated against plant data for the entire power range of operation of the process system (Choi, 1987). Since the open-loop system is unstable, a stabilizing controller is required to allow system operation. The process controller

Table 1
Summary of average SS prediction errors (%)

<table>
<thead>
<tr>
<th>Predictor</th>
<th>Sin. input</th>
<th>Step input</th>
<th>Pulse input</th>
</tr>
</thead>
<tbody>
<tr>
<td>NARX, low noise</td>
<td>4.4</td>
<td>4.9</td>
<td>4.1</td>
</tr>
<tr>
<td>NARX, high noise</td>
<td>5.5</td>
<td>5.3</td>
<td>5.7</td>
</tr>
<tr>
<td>TF, low noise</td>
<td>4.2</td>
<td>4.2</td>
<td>4.0</td>
</tr>
<tr>
<td>TF, high noise</td>
<td>5.3</td>
<td>4.6</td>
<td>5.4</td>
</tr>
<tr>
<td>GF, low noise</td>
<td>4.7</td>
<td>4.7</td>
<td>4.1</td>
</tr>
<tr>
<td>GF, high noise</td>
<td>6.0</td>
<td>5.3</td>
<td>5.7</td>
</tr>
</tbody>
</table>

Table 2
Summary of average MS prediction errors (%)

<table>
<thead>
<tr>
<th>Predictor</th>
<th>Sin. input</th>
<th>Step input</th>
<th>Pulse input</th>
</tr>
</thead>
<tbody>
<tr>
<td>NARX, low noise</td>
<td>6.2</td>
<td>7.7</td>
<td>6.0</td>
</tr>
<tr>
<td>NARX, high noise</td>
<td>7.8</td>
<td>8.2</td>
<td>8.1</td>
</tr>
<tr>
<td>TF, low noise</td>
<td>5.9</td>
<td>6.9</td>
<td>5.7</td>
</tr>
<tr>
<td>TF, high noise</td>
<td>7.8</td>
<td>7.8</td>
<td>7.5</td>
</tr>
<tr>
<td>GF, low noise</td>
<td>5.8</td>
<td>6.4</td>
<td>5.3</td>
</tr>
<tr>
<td>GF, high noise</td>
<td>7.6</td>
<td>7.4</td>
<td>7.4</td>
</tr>
</tbody>
</table>

Fig. 4. Multi-step-ahead response of the first output of the dynamic recurrent predictor to high noise (top, sinusoidal input; middle, step input; bottom, pulse input).
structure used is the one proposed by Menon and Parlos (1992).

The process system has one control input, which is the feed-water flow rate $W_{fw}$, and five disturbance inputs namely, hot-leg temperature $T_{hl}$, primary flow rate $W_{pr}$, steam flow rate $W_{st}$, feed-water temperature $T_{fw}$, and primary pressure $P_{pr}$. The outputs of the process system are three, namely, the water level $L_w$, the secondary pressure $P_{sat}$, and the cold leg temperature $T_{cl}$. Even though the process system has one controlled input, five disturbance inputs and three outputs, only a four-input two-output model has been identified in this study during closed-loop operation. Variations of three disturbances have been neglected. Specifically, the primary loop mass flow rate, the cold leg temperature, and the primary pressure have been neglected, because they tend not to vary radically during most operational transients.

The water level of the process system is strictly regulated between preset limits to: (1) secure that sufficient heat is transferred from the primary side to the secondary side; (2) make sure that excessive moisture, which could damage the turbines, does not carry over; (3) prevent hydro-dynamic instabilities in the feed-water pipes, which are due to the water hammer. If the process water level is not regulated well within preset limits, a plant shut-down is initiated by the plant protection system. The water level control problem is complicated by reverse thermal dynamic (non-minimum phase) effects known as “shrink” and “swell”, which occur during plant transients and are more prominent at start-up/low power operation. Because of the presence of steam bubbles in the tube bundle region of the process system, the water level measured in the downcomer region temporarily reacts in a reverse manner to water inventory changes. The water level control problem is also complicated by the existence of process uncertainties and nonlinear phenomena, which cannot be accurately modeled. Material defects, aging, and corrosion are some of the causes of process parameter variations. In addition, the presence of high process and sensor noise make this a highly complex process system.

5.2. Data sets used in modeling

The empirical process predictor developed in this study
consists of six separate sub-models. Three models to predict the water level response at low, medium and high power operation, respectively, and three other models to predict the secondary pressure response also at low, medium and high power operation. The low power range is defined as 0–20%, the medium power range is 20–50%, while the high power range is 50–100% of full power operation. The reason for breaking-up the operating range of the process is the varying degree of modeling complexity for each power range. For example, the non-minimum phase effects are much more pronounced at the low power levels than at any other power level. Moreover, having three separate predictors increases the accuracy of each one in its own range. The power level is determined by the steam-flow rate measurement, one of the three disturbances included as predictor inputs. This allows selection of the appropriate predictor model. Independent training and testing data sets are collected for predictor development and validation. The training set is further divided into two, for use in weight adaptation and in the selection of the number of hidden layer nodes, respectively. None of the testing data are used until the predictor is completely specified.

A combination of steps and ramps at each power range are used as training and testing data set. The sampling interval for all the data collected is 5 s. The following steps were generated for the low power range: 5–10%, 10–15%, and
15–20% of full power. Similarly, the following ramps with positive and negative ramps were generated for the low power range: 5–20%, and 20–5% of full power, with a magnitude rate of 2% of full power per minute. The testing data consist of a ramp decrease from 15 to 5% of full power at the rate of 2% of full power per minute. The total number of samples for the low power range is 2500. For the low power, the process water level and steam pressure predictors required approximately 10,000 and 14,000 cycles for convergence, respectively.

The following steps were generated for the medium power range: 20–50% of full power with step power increments of 5%. Similarly, the following ramps were generated for the medium power range: 20–50%, and 50–20% of full power with a magnitude rate of 6% of full power per minute. The testing data set consists of a ramp decrease from 45 to 35% of full power at the rate of 6% of full power per minute. The total number of samples for the medium power range is 2530. For the medium power levels, the process water level and steam pressure predictors required approximately 11,000 and 2500 cycles for convergence, respectively.

Finally, for the high power range the following steps were generated: 50–100% of full power with step power increments of 5% of full power. Similarly, the following ramps were generated for the high power range: 50–100%, and 100–50% of full power with a rate magnitude of 8% of full power per minute. The testing data set consists of a ramp decrease from 80 to 60% of full power at a rate magnitude of 8% of full power per minute. The total number of samples for the high power range is 1700. For the high power levels, the process water level and steam pressure predictors required approximately 17,500 and 3500 cycles for convergence, respectively. All the data sets were generated using high process and sensor noise incorporated into the process simulator.

### 5.3. Predictor simulation results

In this section, we present the two predictors developed
for the complex process studied; a polynomial NARX and a dynamic recurrent neural predictor. Results using a recurrent neural predictor trained with TF are not presented because of the unacceptably poor prediction results obtained. For the simulations presented, the average percentage errors are noted in the figure captions.

5.3.1. Polynomial NARX predictor

The polynomial NARX method of Chen and Billings (1989a) was used to develop a predictor for the medium power level range of the process water level. The terms of the polynomial NARX predictor were allowed to be at most cubic with delays up to 100. The reason for allowing such a high order delays is the lack of information regarding the relative order of the process being modeled. This process is in reality an infinite dimensional system described by partial differential equations, and obtaining its relative order is quite difficult. The best performing predictor, as determined

Fig. 8. NARX water level response to a ramp power level decrease from 35 to 20% of full power using a SS predictor ($e = 2.1\%$) and an MS predictor ($e = 5.2\%$).

Fig. 9. UTSG MS neural predictor testing configuration.
by the selection algorithm developed by Chen and Billings (1989b), was the following:

\[ \hat{y}(k) = 0.83y(k-1) - 0.46y^2(k-1)u_1(k-1) + 1.55y^3(k-1), \] (31)

where \( y(\cdot) \) and \( \hat{y}(\cdot) \) stand for the water level measurement and prediction, respectively, while \( u_1(\cdot) \) stands for the feedwater flow rate. This predictor indicates that the only input relevant to the process water level response is the feedwater flow rate. This is only partially true, because both the steam-flow-rate and the feedwater temperature also have a major influence on the process water level response. This result is probably due to the highly complex nature of the process system, as well as the presence of high process and sensor noise in the data.

Fig. 10. UTSG water level transient response for a 5–15% of full power ramp. Top: SS predictor (\( e = 1.81\% \)); MS predictor (\( e = 1.90\% \)). Bottom: steam pressure SS predictor (\( e = 0.07\% \)); MS predictor (\( e = 0.18\% \)).

Fig. 11. UTSG Transient response for a 20–10% of full power ramp: solid line, steam flow rate; dash line: feedwater flow rate.
The predictor of Eq. (31) is used for both SS and MS prediction simulations. Figs. 7 and 8 show the process water level response using the polynomial NARX predictor for a step power level increase from 20 to 25% of full power, and a ramp decrease from 35% of full power level to 20% of full power level, respectively. The NARX predictor minimizes an SS prediction error, and as expected, the SS prediction simulations are more satisfactory than the MS prediction simulations. Fig. 7 demonstrates that the SS prediction water level response is satisfactory for the medium power level range. The water level response in the SS prediction transient shown in Fig. 8 is not satisfactory because of the significant inaccuracy in the peak error. Though, the average error of 2.1% appears acceptable. As seen in Figs. 7 and 8, the polynomial NARX predictor shows poor performance for the MS prediction to both step and ramp power level increase and decrease. This poor MS prediction performance has been consistent for all attempted transients. Because of this no attempt has been made to model the low or high power ranges of operation using the polynomial NARX predictor.

5.3.2. Neural predictor

A recurrent network has been trained with TF and tested with the same transient signals as for the case of the polynomial NARX predictor. The SS prediction results are better than the NARX case for all transients, though they are considered comparable. The MS prediction results are significantly improved compared to the NARX case, though they are still far from being considered acceptable. None of these simulations are presented here, proceeding to the use of dynamic recurrent networks as predictors.

The process system studied is open-loop unstable. Therefore, only closed-loop identification is performed. During the early training stages, while the predictions are highly inaccurate, both the controller and the neural predictors are supplied with the actual process output observations. As the predictor accuracy improves during learning, the output observations are replaced with the predicted process outputs. All testing simulations are performed using predicted values of the process outputs for the predictor and the process controller. The predictor inputs used are those computed by the process controller, and the predictor
disturbances are computed from the assumed power (or load) profile. The predictor testing set-up is depicted in Fig. 9.

The procedure described in Section 4 is used to train dynamic recurrent network predictors. From this procedure, it was determined that outputs with one delay and inputs with two delays (including disturbances) are sufficient. Only two of the four inputs included in the model were delayed for the use in the network, namely the feed-water and the steam flow rates. Latest values of the feed-water flow rate and the hot-leg temperatures were used only. The predicted outputs are the water level and the steam pressure of the secondary side. In summary, the neural predictor outputs are the water level and secondary steam pressure predictions at time $k + 1$; whereas the predictor inputs are water level and secondary steam pressure predictions at time $k + p - 1$, the feed-water flow rate and steam flow rate estimates at times $k + p - 1$, $k + p - 2$ and $k + p + 3$, and feed-water flow rate temperature and hot-leg temperature estimates at time $k + p - 1$. The individual weight update mode is used. The search for the number of hidden nodes resulted in a single hidden layer with four nodes for all the six neural predictors developed. None of the testing data is used in the determination of the number of hidden layer nodes.

The resulting dynamic recurrent neural predictors have revealed good performances for both MS and SS predictions at all power levels. To confirm this finding, extensive simulation tests have been performed for validating the process predictors at all power levels. Figs. 10–16 depict the process water level and secondary pressure response to different transients using the neural predictors and the process simulator. Furthermore, Fig. 11 depicts the process input, the feedwater flow rate, and one of the disturbances, the steam flow rate for the transient corresponding to Fig. 12. The other two disturbances used in the process model are not depicted in this figure because they are both set-points proportional to the steam flow rate. It should be noted that each transient required at least 1 h of computer time to generate using the process simulator, whereas the same transients required only a few seconds to generate using the neural predictors.

The low power range predictors were the most difficult to develop compared to the medium and high power ranges. This is because at the low power range the signal to noise

![Fig. 13. UTSG transient response for a 35–45% of full power ramp. Top: water level SS predictor ($e = 0.37\%$); MS predictor ($e = 0.70\%$). Bottom: steam pressure SS predictor ($e = 0.11\%$); MS predictor ($e = 0.12\%$).](image-url)
(SNR) ratio of the sensed variables is very low compared to the medium and high power level ranges. Figs. 10 and 12 present simulations at the low power level range. These figures demonstrate that even though the performance of the neural predictor at low power levels is acceptable, the water level predictor response is poorer than the steam pressure predictor response. This is attributed to the high process and sensor noise in the water level measurements and to the pronounced non-minimum phase effects in the water level response, which is very significant at low power levels.

The predictors for the medium power range revealed a good performance for all of the simulated transients. Figs. 13 and 14 show the SS and MS prediction simulations in the medium power levels.

The predictors for the high power levels also performed well for all the simulated transients. Figs. 15 and 16 show SS and MS prediction simulations in the high power levels. The responses for the steam pressure are as good as the responses for the water level. This is attributed to the lack of non-minimum phase effects at higher power levels. Note that the average prediction errors for these transients are lower than the ones for the medium power levels, which are lower than the ones for the low power levels. This is to be expected because the SNR at high power levels is about 13 times higher than the SNR at low power levels. Also, the SNR at medium power levels is four times higher than the SNR at low power levels.

6. Discussion and conclusions

The objective of this research study is to present a method for developing predictors in complex process systems, capable of performing accurate MS and SS predictions. A model with accurate MS and SS predictive capabilities finds applications in model predictive process control, and more recently in condition monitoring and fault diagnosis systems. Use of conventional methods for predictive modeling, such as physics-based models, has many disadvantages. These include potential inaccuracies to drifts, inability to incorporate aging, and wear-and-tear effects, development and execution cost. On the contrary, empirical or black-box predictive models combined with the on-line learning capabilities of neural networks could be quite effective in tracking deviations from the nominal dynamics of many
systems. Deviations, such as aging and wear-and-tear effects, are commonly encountered throughout the useful life of many systems.

The proposed neural predictor is based on a dynamic recurrent neural network. In such a network, in addition to the local hidden layer node feedback, the outputs of the network are fed back to itself via tapped delays. It is argued that the network architecture used in the predictor represents an empirical state-space representation that is neither unique nor minimal, requiring as inputs more than just the latest output predictions and observations. A learning algorithm is presented which minimizes the process output MS prediction error. The algorithm is based on the concept of a dynamic gradient descent. In the learning algorithm development, an approximation is made by which all network output predictions present in the input layer, except the latest, are replaced by their respective observations. This assumption enables the derivation of a recursive and computationally efficient algorithm. The developed predictors exhibit acceptable MS prediction accuracy to data unseen during training.

A relatively simple artificial problem is studied demonstrating the comparable performance of three different predictors. As the complexity of the dynamics to be predicted increases, the relative benefit of the proposed method becomes more obvious. The proposed predictor development method is tested on a real-world complex critical process system. This process system is open-loop unstable, exhibiting non-minimum phase response. Therefore, only closed-loop identification can be performed. The developed predictors exhibit good performance for MS and SS prediction simulations at all power level ranges. To confirm this finding extensive simulation tests were performed for validation purposes. Modeling the low power range was more difficult than modeling the medium and high power ranges. This is due to the reverse thermal-hydraulic dynamic effects, which are more pronounced at the low power levels, and it is also due to the high process and sensor noise, also very significant at low power levels. As a result, the performance of the predictors at low power levels is not as good as at medium and high power levels, though it is acceptable. For all the simulations performed, the predicted response for the secondary steam pressure was always better than the response for the process water level. This is due to the fact that the secondary steam pressure is not subject to the reverse dynamic effects as is the process.
Overall, the predictors performed very well throughout the examined process operating range, and for various process and sensor noise levels. The conclusions drawn from this study can be summarized as follows.

- In attempting to obtain accurate MS predictions for highly complex systems, use of a dynamic recurrent neural network appears more effective than the use of either a recurrent or a dynamic architecture with a feedforward network. This has been demonstrated with the case studies of a simple and a complex process.

- The increased complexity and computational cost associated with the learning algorithm derived for the dynamic recurrent neural network appears only justified in significantly complex MS prediction problems. For less complex problems with low noise levels in the observations, either TF training of a recurrent network or training a dynamic (feedforward) network results in satisfactory SS and MS predictors. This has been, again, demonstrated through the two case studies presented.

- The excessive noise associated with the operation of the complex process system studied at low power levels did not prevent the development of an accurate MS predictor. In fact, the developed neural predictor is accurate enough to operate satisfactorily in a closed-loop configuration with the process controller. This configuration was tested for all allowed operational transients of the process. As expected, however, with decreased process and sensor noise the predictor accuracy and, therefore, the closed-loop predictor transient performance is improved.

Acknowledgements

The authors acknowledge the financial support provided to Texas A & M University by the US Department of Energy grant DE-FG07-89ER12893 and the NASA Johnson
Space Center grant NAG#9-347. Dr Atiya would like to acknowledge the support of NSF’s Engineering Research Center at Caltech.

References


