

An Algorithmic Approach to Adaptive State Filtering Using Recurrent Neural Networks

Alexander G. Parlos, *Senior Member, IEEE*, Sunil K. Menon, *Member, IEEE*, and Amir F. Atiya, *Senior Member, IEEE*

Abstract—On-line estimation of variables that are difficult or expensive to measure using known dynamic models has been a widely studied problem. Applications of this problem include time-series forecasting, process control, parameter and state estimation, and fault diagnosis. In this paper, practical algorithms are presented for adaptive state filtering in nonlinear dynamic systems when the state equations are unknown. The state equations are constructively approximated using neural networks. The algorithms presented are based on the two-step prediction-update approach of the Kalman filter. However, unlike the Kalman filter and its extensions, the proposed algorithms make minimal assumptions regarding the underlying nonlinear dynamics and their noise statistics. Nonadaptive and adaptive state filtering algorithms are presented with both off-line and on-line learning stages. The proposed algorithms are implemented using feedforward and recurrent neural network and comparisons are presented. Furthermore, extended Kalman filters (EKFs) are developed and compared to the filter algorithms proposed. For one of the case studies, the EKF converges but results in higher state estimation errors than the equivalent neural filters. For another, more complex case study with unknown system dynamics and noise statistics, the developed EKFs do not converge. The off-line trained neural state filters converge quite rapidly and exhibit acceptable performance. On-line training further enhances the estimation accuracy of the developed adaptive filters, effectively decoupling the eventual filter accuracy from the accuracy of the process model.

Index Terms—Adaptive state filtering, dynamic networks, extended Kalman filters (EKFs), nonlinear state filtering, recurrent networks.

I. INTRODUCTION

ACCURATE on-line estimates of critical system states and parameters are needed in a variety of engineering applications, such as condition monitoring, fault diagnosis, and process control. In these and many other applications it is required to estimate a system variable which is not easily accessible for measurement, using only measured system inputs and outputs. This

is the well-known state estimation problem,¹ which has been traditionally applied to problems such as tracking and guidance of airborne objects, state control, and time-series forecasting [11], [13], [16], [39]. Parameter estimation is also of significant engineering importance, because in many real-world problems accurate values of physical system parameters are not known *a priori* for use in tasks such as control, monitoring, or fault diagnosis. Parameter estimation can be formulated as a state estimation problem, by considering the parameters of interest as additional states [23], [24], [40]. Beyond adaptive control, one of the main applications of parameter estimation is the detection of faults and the monitoring of system condition. By detecting estimated parameter drifts and/or abrupt changes in system parameters, the onset of a fault can be detected in a timely manner [12], [30], [35]. A review of the relevant literature reveals that available parameter and state estimation algorithms are not currently in widespread use in industrial or commercial applications. Rather, they have been limited to defense and other aerospace applications which are customized in nature.

Since the development of the well-known Kalman filter (KF) approach for state estimation in linear systems [11], [21], this method has been widely studied in the literature and applied to many problems. The KF is a recursive estimation approach, where state estimates are obtained in two distinct steps. In the first step, a one-step-ahead state prediction is obtained, based on the latest state estimate. In the second step, the state estimate is refined by linearly combining the state prediction obtained in the first step, with the new output measurements. The KF has been extended to nonlinear systems. One such extension, called the extended Kalman filter (EKF) [11], has experienced a lot of interest from researchers but it has found few industrial applications [20]. In the EKF approach, the state prediction is obtained in the first step using nonlinear state propagation based on the available model. In the second step, the newly received output measurements are combined with the state prediction, still in a linear fashion. As a result the state estimate is effectively a linear combination of all output measurements. In both the KF and the EKF, the model of the system used in the predictor is assumed to be perfectly known, along with the statistics of the process and sensor noise entering the system. These assumptions severely restrict the applicability of these filters in real-world problems and a more general nonlinear state estimation approach is definitely desirable. The EKF algorithm has been around for over 25 years, but it has only in the last a couple of years that rig-

Manuscript received June 15, 1999; revised October 30, 2000 and May 9, 2001. This work was supported by the U.S. Department of Energy under Grant DE-FG07-89ER12893 and by the NASA Johnson Space Center under Grant NAG 9-347. The work of A. F. Atiya was supported by the NSF Engineering Research Center at California Institute of Technology.

A. G. Parlos is with the Department of Mechanical Engineering, Texas A&M University College Station, TX 77843 USA (e-mail: a-parlos@tamu.edu).

S. K. Menon was with the Department of Mechanical Engineering, Texas A&M University College Station, TX 77843 USA. He is now with the Honeywell Technology Center, MN65-2500, Minneapolis, MN 55418 USA (e-mail: smenon@htc.honeywell.com).

A. F. Atiya is with the Learning Systems Group, Department of Electrical Engineering, California Institute of Technology, Pasadena, CA 91125 USA (e-mail: amir@deep.work.caltech.edu).

Publisher Item Identifier S 1045-9227(01)09513-3.

¹For completeness, we should note that state smoothing, filtering, and prediction are all state estimation problems. The distinction is made on the time instant at which the estimate is obtained compared to the latest available observation.

orous analysis for this algorithm has appeared in the literature [7], [36].

Neural networks have been extensively investigated in the context of adaptive control and system identification [6], [18], [25], [29], but it was more recently that their use has been proposed in state estimation. Uses of neural networks in practical state estimation algorithms has been scarce. One of the first investigation of the filtering applications of neural networks was by Lo who proved that a state filter based on recurrent neural networks converges to the minimum variance filter [26]. Elanayar and Shin used radial basis function neural networks to investigate state estimation problems [10]. Stubberud *et al.* have presented an adaptive form of the EKF which uses a neural network to develop an error dynamics model identified simultaneously with the state estimates [42]. Suykens *et al.* have used the so-called neural state-space representation for nonlinear identification and control, though they did not investigate its application to state estimation [44]. Obradovic used Gaussian radial basis function networks also to predict system outputs and in cases when the states and outputs are related by linear relations, to also predict the system states [31]. Nonlinear finite-memory state estimators using neural networks have also been proposed by Parisini *et al.* [32] and Alessandri *et al.* [1], in related publications. Haykin *et al.* presented an overview of state filtering with neural networks with emphasis on radial basis function networks [17]. Zhu *et al.* proposed the design of an adaptive observer using dynamic neural networks [46] and Habtom investigated the use of the EKF with an identified neural-network model [15]. Lei *et al.* used recurrent neural networks to estimate the state of a chemical process [22], whereas Schenker and Agarwal used neural networks for predictive control of a chemical reactor that involved state estimation [38]. Finally, Stubberud *et al.* used neural networks to implement an EKF, comparing various implementation options [43]. Also recently, the problem of parameter estimation has received the attention of the neural networks community with some new results reported [2].

In this paper, practical algorithms for effective adaptive state filtering are presented for nonlinear dynamic systems. These algorithms are based on the well-known universal approximation properties of feedforward and recurrent neural networks [4], [5], [9], [19]. In an attempt to broaden the applicability of the proposed algorithms, minimal assumptions are placed upon the dynamic structure and the noise characteristics of the underlying dynamics. As such, rigorous analysis of the algorithms usually found in linear filtering settings, or based on linearization arguments, is not a major thrust of this paper. Such an effort will necessitate assumptions that invalidate the original intent of this work, that is its applicability to problems with unknown dynamics. The algorithms are demonstrated in some real-world problems and compared to EKF for effectiveness. This paper makes the following specific contributions.

- Practical algorithms are presented for adaptive state filtering in nonlinear dynamic systems that could be applicable to a broad range of problems. Our formulation is based on the recursive approach of the EKF. However, unlike this formulation, the case of unknown state equations

is considered. Separate neural networks models are developed for each “mapping” involved in the filtering algorithms.

- The effectiveness of the presented algorithms are demonstrated by developing state filters for an artificial nonlinear problem and a real-world complex process encountered in power plants, namely a U-tube steam generator (UTSG). The performance of the developed neural filters is extensively tested and compared to EKF, demonstrating the resulting improvements in performance and broad applicability.

In the next section, we formulate the general nonlinear state filtering problem and describe a conventional solution approach to the problem followed by the constructive approach using neural networks. In Section III, the algorithm for the nonadaptive neural state filter is presented, followed by the algorithm for adaptive neural state filter in Section IV. In Section V, the simulation results demonstrating the effectiveness of the filtering algorithms are presented. Section VI presents some discussion, along with a summary and the conclusions drawn from the paper.

II. NONLINEAR STATE FILTERING

In this section the general nonlinear state filtering problem is formulated. A brief overview of the EKF is presented, along with the approximation-based approach formulated in this paper.

A. Problem Statement

Consider the following system representation in the discrete-time nonlinear state-space form, also known as the *noise representation*

$$\begin{cases} \mathbf{x}(t+1) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) + \mathbf{w}(t) \\ \mathbf{y}(t) = \mathbf{h}(\mathbf{x}(t)) + \mathbf{v}(t) \end{cases} \quad (1)$$

where

$t = 1, 2, \dots$	discrete-time instant;
$\mathbf{y}(t)$	$n \times 1$ output vector of the nonlinear state-space model;
$\mathbf{u}(t)$	$m \times 1$ input vector of the model;
$\mathbf{x}(t)$	$l \times 1$ state vector of the nonlinear model;
$\mathbf{f}(\cdot)$ and $\mathbf{h}(\cdot)$	vector-valued continuous nonlinear functions;
$\mathbf{w}(t)$	process noise, an $l \times 1$ vector;
$\mathbf{v}(t)$	measurement noise, an $n \times 1$ vector.

Further, it is assumed that $\mathbf{w}(t)$ and $\mathbf{v}(t)$ are independent processes. At this stage no assumptions are made regarding implicit or explicit knowledge of $\mathbf{f}(\cdot)$ and $\mathbf{h}(\cdot)$, but that such a representation is an accurate description of the underlying dynamics. In many traditional state filtering approaches, exact knowledge of (1) is assumed and the noise vectors $\mathbf{w}(t)$ and $\mathbf{v}(t)$ are considered zero-mean white Gaussian. Equation (1) becomes the starting point for our development.

The objective of the state filtering problem is to obtain an estimate $\hat{\mathbf{x}}(t)$ for the state $\mathbf{x}(t)$ given input and output observations. In linear state filtering, the notation used to denote this estimate

is important because depending on the chosen filtering method, different optimal estimates are obtained. Nevertheless, in real-world nonlinear state filtering problems the resulting state estimates are not optimal in any sense. Therefore, we use the notation $\hat{\mathbf{x}}(t|t)$ to mean the state estimate at time t , following the update resulting from the measurements $\mathbf{u}(t)$ and $\mathbf{y}(t)$, at time t . This notation should not be interpreted to imply an optimal state estimate in the minimum variance or any other sense.

B. Conventional Method of Solution

Conventional methods of solving nonlinear state filtering problems have almost exclusively been limited to the nonadaptive case where it is assumed that an exact system model is available. Methods for adaptive nonlinear state filtering have almost always been limited to the tuning of the filter gains in an *ad hoc* manner following the design of a nonadaptive filter.

In general, there are two approaches to solving nonlinear (nonadaptive) state filtering problems; one is formulated as a nonlinear least-squares estimation problem, minimizing the error between the measured system output and the predicted system output, whereas the other approach attempts to minimize the variance of the state estimation error. The resulting state filters are known as least-squares and minimum variance filters, respectively. Both approaches require explicit knowledge of the nonlinear system dynamics, though the assumptions placed on the statistics are different.

In nonlinear least-squares estimation no statistical assumptions are made regarding the noise terms because the state estimate is chosen as a “best fit” to the observed measurements in a deterministic sense. The problem is formulated as that of a classical nonlinear parameter optimization, where an objective function is minimized subject to the constraints imposed by the system dynamics. For general nonlinear least-squares problems no closed-form solutions exist and approximations must be obtained. The various approaches used are the gradient descent and conjugate gradients methods. To obtain a recursive nonlinear filter, a sequence of a nonlinear least-squares objective functions must be minimized. This requires further approximations to be introduced. Nonlinear minimum variance estimation, on the other hand, requires explicit knowledge of the noise statistics involved in the system dynamics. The resulting nonlinear state estimate is the conditional mean of the actual state and the resulting state filter is recursive in nature. The main approximation introduced in nonlinear minimum variance estimation is the one involving the system dynamics. There are two major approaches for approximating the nonlinear system dynamics; either by Taylor series expansion or by statistical linearization. Different forms of the Taylor series expansion result in the EKF, the linearized KF, and other high-order filters, such as the iterated EKF [11]. In all of the aforementioned nonlinear state estimation approaches it is assumed that an exact nonlinear model is available. Furthermore, in the case of minimum variance estimation the statistics of the noise processes involved in the system dynamics are assumed known.

The EKF solution to nonlinear state filtering is briefly presented assuming the availability of an exact system model depicted by (1) [11], [14]. The algorithm consists of the following two steps.

Step 1—Prior to Observing the $(t + 1)^{th}$ Sample-Prediction Step: In this step the assumed model $\mathbf{f}(\cdot)$ and $\mathbf{h}(\cdot)$ and the state estimate $\hat{\mathbf{x}}(t|t)$ are used to compute the predicted value of the state and output, $\hat{\mathbf{x}}(t + 1|t)$ and $\hat{\mathbf{y}}(t + 1|t)$, respectively. The *a priori* error covariance matrix is also computed using the state estimate and the Jacobian of $\mathbf{f}(\cdot)$ evaluated at the state estimate. The assumed covariance matrix of the process noise is also used in this step.

Step 2—Following Observation of $(t + 1)^{th}$ Sample-Update Step: Following the arrival of a new measurement $\mathbf{y}(t + 1)$, the state prediction, $\hat{\mathbf{x}}(t + 1|t)$, is updated using a linear combination of the prediction errors (or innovations), $\mathbf{y}(t + 1) - \hat{\mathbf{y}}(t + 1|t)$, resulting in the state estimate $\hat{\mathbf{x}}(t + 1|t + 1)$. The coefficients used to weigh the innovations terms form the elements of the EKF gain matrix. The EKF gain matrix is updated in this step using the Jacobian of $\mathbf{h}(\cdot)$ evaluated at the state prediction, $\hat{\mathbf{x}}(t + 1|t)$, the *a priori* error covariance matrix and the assumed covariance matrix of the measurement noise. Finally, the *a posteriori* error covariance matrix is computed using the updated EKF gain matrix, the *a priori* error covariance matrix and the Jacobian of $\mathbf{h}(\cdot)$ evaluated at the state prediction, $\hat{\mathbf{x}}(t + 1|t)$.

The approximations introduced in the EKF often result in inaccurate state estimates and the EKF frequently suffers from convergence problems. Many alternatives have been proposed, such as square-root filtering to alleviate convergence problems. Additionally, in many circumstances the linearized KF is used. In the linearized KF, the nonlinear system dynamics are linearized about a nominal state trajectory, $\bar{\mathbf{x}}(t)$, instead of the state estimate. The linearized KF filter though has less convergence problems than the EKF, it usually results in less accurate state estimates.

C. Neural Method of Solution

In principle, the proposed neural method of solution appears similar to the EKF, however there are three significant differences.

- The nonlinear functions $\mathbf{f}(\cdot)$ and $\mathbf{h}(\cdot)$ are assumed to be unknown and they are either replaced by a known model $\mathbf{f}_{\text{mod}}(\cdot)$ and $\mathbf{h}_{\text{mod}}(\cdot)$ in the nonadaptive filter, or constructed from input–output measurements in the adaptive filter.
- The functional form of the filter state update equation is allowed to be nonlinear, say $\mathcal{K}(\cdot)$ and it is also constructed from input–output measurements.
- The noise statistics are assumed unknown and they are not explicitly needed in the filter computations.

The networks considered in this paper for approximating the filter functional forms are the feedforward multilayer perceptron (FMLP) and the recurrent multilayer perceptron (RMLP) [33], both “universal approximators”² [4], [5], [9], [19]. In the next section learning algorithms are presented for training nonlinear filters based on these neural networks. Both the nonadaptive and the adaptive filtering cases are presented. Specifically, by approximating the functions $\mathbf{f}(\cdot)$, $\mathbf{h}(\cdot)$ and $\mathcal{K}(\cdot)$ using neural

²Strictly speaking, the proof for an FMLP is known, though we are not aware of an equivalent proof for RMLP networks.

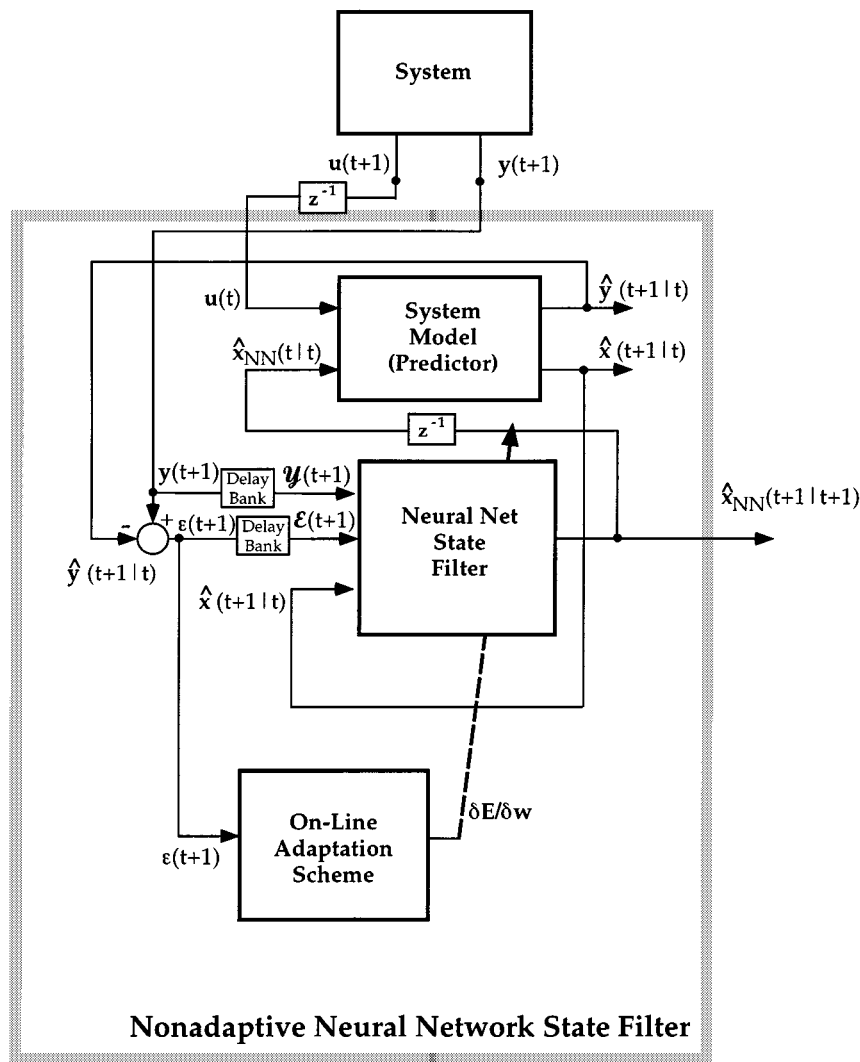


Fig. 1. Block diagram of the nonadaptive neural-network state filter.

networks, nonlinear least-squares filtering algorithms can be developed for use in determining the filter structure and parameters.

III. NONADAPTIVE NEURAL STATE FILTER

In this section the case of a nonadaptive neural state filter is considered. In view of the complexity of the problems addressed, the nonadaptive formulation is of academic interest because in many cases of interest exact nonlinear models are not available. However, the nonadaptive formulation helps setup the framework for the more complex adaptive case. In nonadaptive state filtering a deterministic system model is assumed available. Such a model could be in the form of a complex simulator or an analytic model. The system model can be simulated off-line and during such simulations the system states can be computed and used for off-line filter design. On the other hand, during on-line filtering operations the (current and past) values of system states are not available. Fig. 1 shows a block diagram of the nonadaptive state filter configuration, with on-line filter tuning.

A. Filter Equations

The state filter utilizing neural networks assumes that a deterministic system model is available with the structural form of (1). It is assumed that given a value of the state $\mathbf{x}_{\text{mod}}(t)$ and the input $\mathbf{u}(t)$, the state $\mathbf{x}_{\text{mod}}(t+1)$ and output $\mathbf{y}_{\text{mod}}(t+1)$ can be computed using the deterministic model $\mathbf{f}_{\text{mod}}(\cdot)$ and $\mathbf{h}_{\text{mod}}(\cdot)$, respectively. Neural networks are used to model some of the functional relations entering a recursive filter. This constructive approach to state filtering can be considered *nonadaptive* because a system model is assumed available and no further identification is performed. It is further assumed that the system inputs and outputs are measured.

As in the case of the EKF, our approach is also recursive. This means that each updated estimate of the state is computed in terms of the previous state estimate and the new observations. As such, it consists of a prediction and an update step, much like the EKF. However, unlike the EKF, where the update step is a linear combination of the latest innovations, in our approach the update step can take a general nonlinear form and include past innovations. Since an analytical formulation and solution for a nonlinear state filter is very difficult, even for known specific

forms of $\mathbf{f}_{\text{mod}}(\cdot)$ and $\mathbf{h}_{\text{mod}}(\cdot)$, a neural-network approach is followed to learn the filter relation $\mathcal{K}(\cdot)$ from data.

To obtain a state estimate, $\hat{\mathbf{x}}(t+1|t+1)$, using a nonlinear system model of the form given by (1), the following prediction-update algorithm is proposed.

Step 1—Prior to Observing the $(t+1)$ th Sample-Prediction Step:

$$\begin{cases} \hat{\mathbf{x}}(t+1|t) = \mathbf{f}_{\text{mod}}(\hat{\mathbf{x}}_{NN}(t|t), \mathbf{u}(t)) \\ \hat{\mathbf{y}}(t+1|t) = \mathbf{h}_{\text{mod}}(\hat{\mathbf{x}}(t+1|t)) \end{cases} \quad (2)$$

where $\hat{\mathbf{x}}(t+1|t)$ and $\hat{\mathbf{y}}(t+1|t)$ are the system state and output predictions, respectively, and where $\hat{\mathbf{x}}_{NN}(t|t)$ is the neural state estimate.

Step 2—Following Observation of $(t+1)$ th Sample-Update Step:

$$\hat{\mathbf{x}}_{NN}(t+1|t+1) = \mathcal{K}_{NN}(\hat{\mathbf{x}}(t+1|t), \mathcal{Y}(t+1), \mathcal{E}(t+1)) \quad (3)$$

where $\mathcal{Y}(t+1)$ is a vector containing present and past system output measurements and $\mathcal{E}(t+1)$ is a vector containing the present and past innovations terms. These vectors are defined as follows:

$$\begin{cases} \mathcal{Y}(t+1) \equiv [\mathbf{y}(t+1), \mathbf{y}(t), \dots, \mathbf{y}(t-n_y+1)]^T \\ \mathcal{E}(t+1) \equiv [\epsilon(t+1), \epsilon(t), \dots, \epsilon(t-n_e+1)]^T \end{cases} \quad (4)$$

where

$$\epsilon(t+1) \equiv \mathbf{y}(t+1) - \hat{\mathbf{y}}(t+1|t) \quad (5)$$

is the innovations term defined in the EKF, $\mathcal{K}_{NN}(\cdot)$ is a nonlinear function equivalent in functionality to the EKF gain and where n_y and n_e are the number of past output and error terms used in the filter, respectively. Equations (2) and (3) represent the nonadaptive state filter equations.

In contrast to the EKF algorithm, a prediction and an update equation for the error covariance matrix is not included. It is quite natural to augment the filter (2) and (3), with functional forms for the *a priori* and *a posteriori* error covariance matrices. During actual operation of the developed filters such estimates of the error covariance matrix could become crucial in determining filter performance because of the lack of an actual state measurements.

In the nonadaptive filter there is only one network to be constructed, $\mathcal{K}_{NN}(\cdot)$. Furthermore, as seen by (3), the mapping performed by this function is static. Therefore, an FMLP network should be sufficient for this approximation. Two approaches are used in the filter training, teacher forcing (TF) and global feedback (GF). If the input layer of a network contains autoregressive terms or other variables that are not easily measured on-line, as in the case of state filtering, training can proceed in two ways. In training with TF, only the samples available in the training set are used as network inputs. Whereas, in training with GF past network predictions are used as networks inputs instead of training samples. In other words, TF trains a neural network statically whereas GF results in dynamic training. In this study, the filter network training procedure has been partitioned into two phases. In the

first phase, the so-called off-line phase, the system is simulated numerically by providing the system input signals that will typically be encountered during its normal operation. For such a phase the state vectors are assumed known, obtained from the available model. The resulting training setup consists of an FMLP network trained using TF. In the second phase, the *on-line* phase, the system is assumed to be in real operation. Input and output data are collected and further training is performed to fine tune the parameters of the filter. For the on-line phase the states are not assumed available. In this stage an FMLP network is further trained using GF. Other considerations in network design, such as prevention of overfitting, assignment of network nodes, etc., are not discussed here because they follow standard practices [18], [28].

B. Off-Line Training

For the off-line training case a finite set of state values is assumed available, constructed either by model simulations or off-line measurements. The network error function is defined as follows:

$$E \equiv \sum_{t=0}^{NP} E(t) \equiv \sum_{t=0}^{NP} \sum_{k=1}^{\ell} [\hat{x}_{NN,k}(t|t) - x_{\text{target},k}(t)]^2 \quad (6)$$

where

$x_{\text{target},k}(t)$	k^{th} component of the target vector $\mathbf{x}_{\text{target}}(t)$;
$\hat{x}_{NN,k}(t t)$	k^{th} component of the state estimate vectors $\hat{\mathbf{x}}_{NN,k}(t t)$;
NP	number of training examples;
ℓ	number of states to be estimated.

Because state measurements are available, the inputs to the filter network $\mathcal{K}_{NN}(\cdot)$, i.e., $\hat{\mathbf{x}}(t+1|t)$, $\mathcal{Y}(t+1)$, $\mathcal{E}(t)$, are all known. Therefore, a TF strategy is adopted to train the filter network. The learning algorithm could be the simple backpropagation method applied to the network represented by the function $\mathcal{K}_{NN}(\cdot)$, with inputs $\hat{\mathbf{x}}(t+1|t) \equiv \mathbf{f}_{\text{mod}}(\mathbf{x}_{\text{target}}(t), \mathbf{u}(t))$, $\mathcal{Y}(t+1)$ and $\mathcal{E}(t)$, with output $\hat{\mathbf{x}}_{NN}(t+1|t+1)$ and with desired output $\mathbf{x}_{\text{target}}(t+1)$. Any variant of the backpropagation algorithm is also appropriate. Because any such approach is based on a gradient descent algorithm, the off-line training will converge at least to a local minimum [18]. Further assume that a large enough training set is assembled, that the network is large enough and that it converged to the global minimum. Then it can be argued that the off-line state estimate will asymptotically converge to the expected value of the target state, given the current and previous outputs and previous state estimates [37]. In interpreting these qualitative arguments about the convergence properties of the off-line algorithm, one should consider the current algorithmic limitations in designing neural networks for dynamic systems, with respect to their generalization properties.

C. On-Line Training

For the on-line case the states are unknown and the training algorithm becomes more complex. Observing (3) reveals that the right-hand side contains a number of past predictions, observations and innovations. These in turn depend on the network outputs at previous time steps. This suggests a recurrent-type learning algorithm, even if using the simple FMLP architecture

for the filter network. An algorithm based on the forward propagation approach of Williams and Zipser is used in this study [18], [45]. Although such an approach is slower than other approaches, such as the backpropagation-through-time (BTT), it has the comparative advantage of being easily adapted for use as an on-line method. In addition, in the event that training must be continued while the system is in operation, the on-line training feature is more desirable. A recently developed algorithm that is considerably faster than existing on-line techniques would be appropriate for this phase of the training [3].

Since the states are unknown, the error here is defined in terms of the system outputs as follows:

$$E(t+1) \equiv \sum_{k=1}^n [\hat{y}_k(t+1|t) - y_{\text{target},k}(t+1)]^2 \quad (7)$$

where

$$\begin{array}{ll} y_{\text{target},k}(t+1) & k^{\text{th}} \text{ component of } \mathbf{y}_{\text{target}}(t+1); \\ \hat{y}_k(t+1|t) & k^{\text{th}} \text{ components of } \hat{\mathbf{y}}(t+1|t); \\ n & \text{number of outputs included in the training.} \end{array}$$

In the following notation, for any two vectors $\mathbf{a} = (a_1, \dots, a_n)^T$ and $\mathbf{b} = (b_1, \dots, b_m)^T$, $\partial\mathbf{a}/\partial\mathbf{b}$ means the matrix with $\partial a_i/\partial b_j$ as its (i, j) th component. The error gradients can be obtained by using the chain rule as follows:

$$\frac{\partial E(t+1)}{\partial \mathbf{w}} = 2 [\hat{\mathbf{y}}(t+1|t) - \mathbf{y}_{\text{target}}(t+1)]^T \frac{\partial \hat{\mathbf{y}}(t+1|t)}{\partial \mathbf{w}} \quad (8)$$

where \mathbf{w} contains all of the network weights and biases. The gradient $\partial \hat{\mathbf{y}}(t+1|t)/\partial \mathbf{w}$ can be obtained in terms of the gradient of the state with respect to \mathbf{w} , by differentiating the predictor (2) with respect to \mathbf{w} , resulting in

$$\frac{\partial \hat{\mathbf{y}}(t+1|t)}{\partial \mathbf{w}} = \left(\frac{\partial \mathbf{h}_{\text{mod}}}{\partial \hat{\mathbf{x}}(t+1|t)} \right) \left(\frac{\partial \mathbf{f}_{\text{mod}}}{\partial \hat{\mathbf{x}}_{NN}(t|t)} \right) \left(\frac{\partial \hat{\mathbf{x}}_{NN}(t|t)}{\partial \mathbf{w}} \right). \quad (9)$$

The state gradient vector $\partial \hat{\mathbf{x}}_{NN}(t|t)/\partial \mathbf{w}$ can be obtained in terms of the state gradient vectors at times $t' < t$, by repeated application of the chain rule. By differentiating (3) (with t substituted in place of $t+1$), the following is obtained:

$$\begin{aligned} \frac{\partial \hat{\mathbf{x}}_{NN}(t|t)}{\partial \mathbf{w}} &= \frac{\partial \mathcal{K}_{NN}}{\partial \mathbf{w}} \\ &+ \left(\frac{\partial \mathcal{K}_{NN}}{\partial \hat{\mathbf{x}}(t|t-1)} \right) \left(\frac{\partial \mathbf{f}_{\text{mod}}}{\partial \hat{\mathbf{x}}_{NN}(t-1|t-1)} \right) \\ &\times \left(\frac{\partial \hat{\mathbf{x}}_{NN}(t-1|t-1)}{\partial \mathbf{w}} \right) \\ &+ \sum_{i=0}^{n_e} \left(\frac{\partial \mathcal{K}_{NN}}{\partial \hat{\mathbf{y}}(t-i|t-1-i)} \right) \\ &\times \left(\frac{\partial \hat{\mathbf{y}}(t-i|t-1-i)}{\partial \mathbf{w}} \right) \end{aligned} \quad (10)$$

where $\partial \mathbf{y}(t-i)/\partial \mathbf{w}$ was set to zero. The gradients of $\mathcal{K}_{NN}(\cdot)$ with respect to \mathbf{w} , $\hat{\mathbf{x}}(t+1|t)$ and $\hat{\mathbf{y}}(t-i|t-1-i)$ depend on the specific architecture of the network used and for the FMLP network they can be obtained using a backpropagation-type procedure. For recurrent architectures, such as the RMLP network, an approach similar to the one presented in [33] can be used. The derivatives of $\mathbf{f}_{\text{mod}}(\cdot)$ and $\mathbf{h}_{\text{mod}}(\cdot)$ with respect to the states can

be obtained numerically or even analytically, if analytical expressions for $\mathbf{f}_{\text{mod}}(\cdot)$ and $\mathbf{h}_{\text{mod}}(\cdot)$ are available. Equations (9) and (10) summarize the recursion used to obtain the error gradients involved in the state filter. Once the error gradient is computed, the weight update is performed using a gradient descent rule.

D. Off-Line Versus On-Line Training

It should be noted here that both the off-line and on-line training stages are complementary and essential for training the network. Applying only off-line training will usually not be sufficient. This can be seen by observing (3) and focusing on the input $\hat{\mathbf{x}}(t+1|t)$. For the off-line case, $\hat{\mathbf{x}}(t+1|t)$ is given as $\mathbf{f}_{\text{mod}}(\mathbf{x}_{\text{target}}(t), \mathbf{u}(t))$ and the neural network essentially attempts to infer the value of $\mathbf{x}_{\text{target}}(t+1)$ given the values of $\mathbf{x}_{\text{target}}(t)$ and $\mathbf{y}(t+1)$. On the other hand, for the on-line case the network attempts to infer $\mathbf{x}_{\text{target}}(t+1)$ given $\mathbf{y}(t+1)$ and the previous estimate $\hat{\mathbf{x}}_{NN}(t|t)$, which is not exactly the same as $\mathbf{x}_{\text{target}}(t)$. Therefore, the on-line case is the more realistic and indeed the more accurate approach, allowing the filter to operate in closed-loop form. The off-line stage can be viewed as the initial training phase used to obtain a good starting point for the on-line training stage. The ease of implementing off-line training (it is merely a standard backpropagation procedure), gives us an excellent opportunity to achieve such a relatively good starting point. Further improvements to the state estimates can be made using on-line learning. The major drawback of the on-line learning is the lack of guaranteed convergence of the associated algorithm.

IV. ADAPTIVE NEURAL STATE FILTER

Adaptive state filters are used when a system model must be identified. In the EKF method for state filtering, the functions $\mathbf{f}(\cdot)$ and $\mathbf{h}(\cdot)$ are assumed known. Currently there are no practically viable algorithms for designing adaptive state filters for nonlinear dynamic systems.

A. Filter Equations

In this section, no system model is assumed known for use in the predictor step of the filter. Rather, a finite set of input, state, and output measurements is assumed available. Therefore, neural networks are used to develop approximations of the functional forms entering the predictor step of the filter equations. This approach to state filtering is considered *adaptive* because an empirical system model must be constructed first. The state-space representation of (1) can be rewritten in the *innovations* representation as follows:

$$\begin{cases} \hat{\mathbf{x}}(t+1|t) = \mathbf{f}_{\text{innov}}(\hat{\mathbf{x}}(t|t-1), \mathbf{u}(t), \boldsymbol{\mathcal{E}}(t)) \\ \hat{\mathbf{y}}(t+1|t) = \mathbf{h}_{\text{innov}}(\hat{\mathbf{x}}(t|t-1), \mathbf{u}(t), \boldsymbol{\mathcal{E}}(t)) \end{cases} \quad (11)$$

where $\mathbf{f}_{\text{innov}}(\cdot)$ and $\mathbf{h}_{\text{innov}}(\cdot)$ are nonlinear functions related to functions $\mathbf{f}(\cdot)$ and $\mathbf{h}(\cdot)$ of the noise representation [23].

The purpose of the *innovations* term $\boldsymbol{\mathcal{E}}(t)$, as defined earlier, is to account for stochastic effects and modeling uncertainties not predicted by the filter predictor. Therefore, in the *innovations* form of the state-space representation, the error in the state prediction $\hat{\mathbf{x}}(t|t-1)$ is compensated by the *innovations* term,

$\mathcal{E}(t)$. The transition from the nonlinear *innovations* representation to the *prediction-update* form is made by considering this crucial role of the *innovations* term.

In the two-step prediction-update state filter, the update step must use the most current system output measurements in the form of innovations, to compensate for stochastic and/or modeling inaccuracies in the prediction step. Therefore, in the prediction step only the most relevant signals are required for computing $\hat{\mathbf{x}}(t+1|t)$, that is the updated state value $\hat{\mathbf{x}}(t|t)$, the measured system inputs $\mathbf{u}(t)$, and the present and past output predictions, $\hat{\mathbf{y}}(t|t-1)$, or output measurements $\mathcal{Y}(t)$. Since the updated state value has already been compensated by the innovations term, the use of both $\mathcal{Y}(t)$ and $\hat{\mathbf{y}}(t|t-1)$ in obtaining the state/output prediction is redundant. Therefore, a choice can be made regarding the use $\mathcal{Y}(t)$ over $\hat{\mathbf{y}}(t|t-1)$ in the prediction equations and vice versa. In this study the latter of the two variables is used. This observation about the innovations term results in a predictor form that utilizes the most recent state information and this predictor can be directly used in the filter equations. Furthermore, the nonlinear functions $\mathbf{f}_{\text{innov}}(\cdot)$ and $\mathbf{h}_{\text{innov}}(\cdot)$ of the predictor can be constructed using neural networks, as in the neural filter function $\mathcal{K}_{NN}(\cdot)$ of the nonadaptive state filter.

The adaptive state filter for $\hat{\mathbf{x}}(t+1|t+1)$ can now be formulated using the following two steps.

Step 1—Prior to Observing the $(t+1)$ th Sample-Prediction Step:

The state and output predictor values are obtained using the following equations:

$$\begin{cases} \hat{\mathbf{x}}_{NN}(t+1|t) = \mathbf{f}_{NN}(\hat{\mathbf{x}}_{NN}(t|t), \mathbf{u}(t), \hat{\mathbf{y}}_{NN}(t|t-1)) \\ \hat{\mathbf{y}}_{NN}(t+1|t) = \mathbf{h}_{NN}(\hat{\mathbf{x}}_{NN}(t|t), \mathbf{u}(t), \hat{\mathbf{y}}_{NN}(t|t-1)) \end{cases} \quad (12)$$

where $\hat{\mathbf{x}}_{NN}(t+1|t)$ and $\hat{\mathbf{y}}_{NN}(t+1|t)$ are the neural state and output predictions and where $\hat{\mathbf{y}}_{NN}(t|t-1)$ is the vector containing the current and past output predictor responses.

Step 2—Following Observation of the $(t+1)$ th Sample-Update Step:

The state prediction is updated using

$$\hat{\mathbf{x}}_{NN}(t+1|t+1) = \mathcal{K}_{NN}(\hat{\mathbf{x}}_{NN}(t+1|t), \mathcal{Y}(t+1), \mathcal{E}_{NN}(t+1)) \quad (13)$$

where the vectors $\mathcal{Y}(\cdot)$, $\mathcal{E}_{NN}(\cdot)$ are defined similarly to the nonadaptive filter case. It should be noted that the output predictor form is not identical to the standard EKF formulation. It has been our experience that the present formulation results in improved state estimates, as compared to the standard EKF formulation. At this time, we cannot offer an explanation for this. A schematic diagram of the adaptive state filtering method is shown in Fig. 2, where the on-line adaptation of the state and output predictors and the state update network is shown.

For the adaptive neural filter case, there are three neural networks to be trained, represented by the functions $\mathbf{f}_{NN}(\cdot)$, $\mathbf{h}_{NN}(\cdot)$ and $\mathcal{K}_{NN}(\cdot)$. The filter update network $\mathcal{K}_{NN}(\cdot)$ is treated similarly as in the case of the nonadaptive filter. It represents a static mapping and it is approximated by an FMLP

trained using TF and GF for the off-line and TF for the on-line stages. The state and output predictor networks represent dynamic mappings and they are approximated by either FMLP or RMLP networks. They are trained using TF and GF for the off-line and TF for the on-line stages also. It has been our experience that as the complexity of the state filtering problem increases, RMLP networks are more effective in approximating the two predictors. Furthermore, training predictors using GF results in improved generalization capability.

B. Off-Line Training

It is assumed that some reliable state measurements are available from experiments done on the system itself or a system model, or that even other off-line signal processing methods are available for extracting the state values from input and output measurements, for example using spectral estimation techniques. Such an assumption is quite realistic because in a number of real-world systems a limited number of measurements can be collected for variables that might otherwise be very costly to monitor and sense. For example, in process systems, laboratory analysis is performed routinely to determine the composition of produced compounds, a rather expensive sequence to be performed continuously for on-line measurements. Similarly, in many electromechanical devices speed measurements might not be available due to the cost associated with speed sensors. Nevertheless, hand-held tachometers can be utilized during an initial setup period to collect speed information. Additionally, off-line signal processing methods can be used to accurately infer speed information from input and output measurements.

The off-line training is divided into two phases. The difference between the two phases is in the interaction of the three networks during training. In the former the networks are trained separately using TF, whereas in the latter they are coupled and trained using GF. In this first phase, the inputs to the output and state predictor networks are the present and past inputs of the system, $\mathbf{u}(t)$, the present and past output observations, $\mathcal{Y}_{\text{target}}(t)$ and the state value, $\mathbf{x}_{\text{target}}(t)$, as obtained by off-line experiments, simulations, or other signal processing procedures. The inputs to the state update network are the present and past outputs of the system, $\mathcal{Y}_{\text{target}}(t+1)$, the present and past *innovations* terms, $\mathcal{E}_{\text{target}}(t+1)$ and the state value, $\mathbf{x}_{\text{target}}(t)$, obtained as previously discussed. This is the TF phase of the off-line training. Once the learning process reaches an acceptable level of accuracy then GF is used to minimize the multi-step-ahead prediction (MSP) error [34]. In the GF phase of the training process, the two predictors and the state update network are developed in tandem, by using the response of one network to improve the predictive response of the other, until both predictors and the state update network produce acceptably accurate responses. The output and state predictors are constructed using either RMLP or FMLP networks. For simple state filtering problems use of FMLP networks for the output and/or state predictors appears sufficient. As the complexity of the filtering problem increases the use of RMLP networks becomes essential. Irrespective of the difficulty of the filtering problem, the FMLP network architecture is used in constructing the state update network because of its static nature.

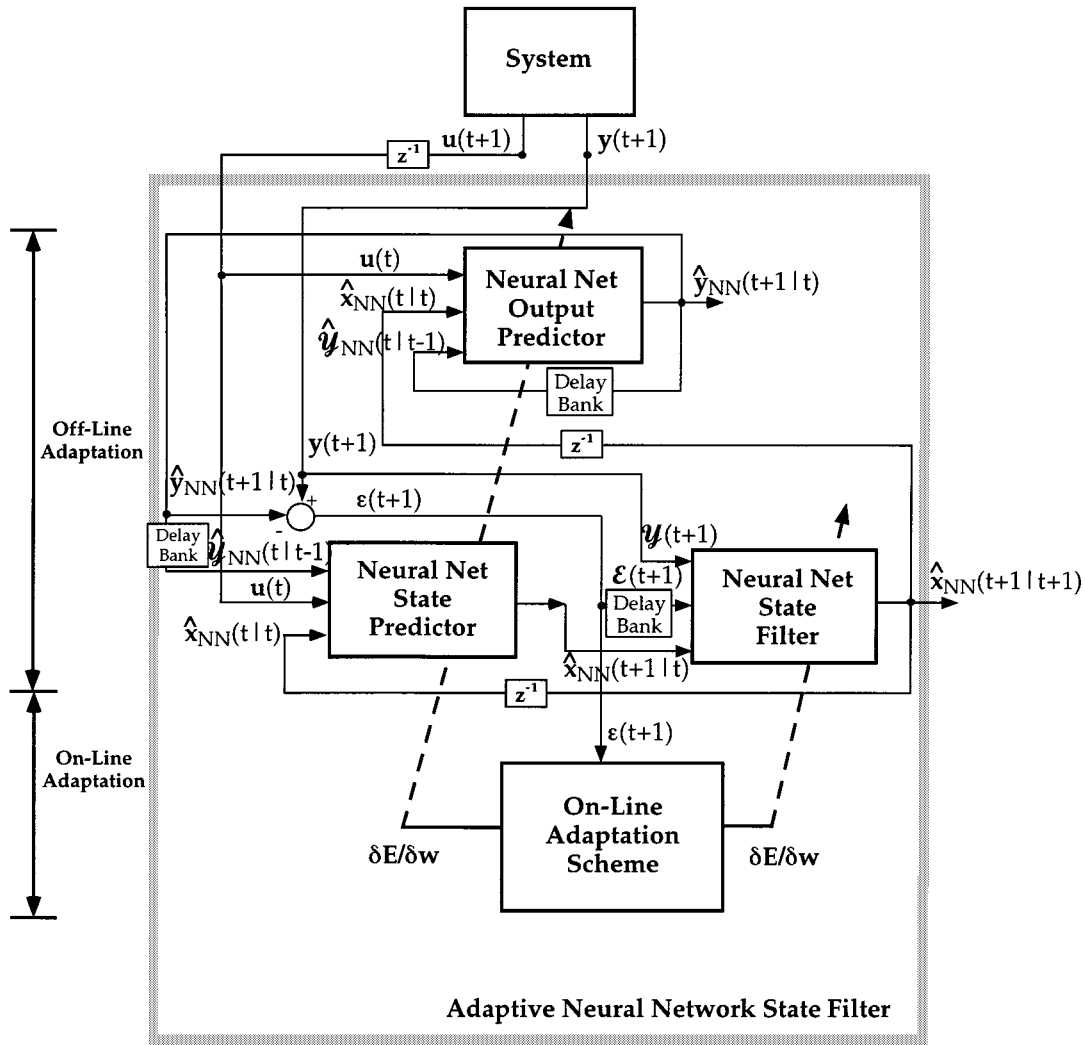


Fig. 2. Block diagram of the adaptive neural network state filter.

In the first phase of the off-line training process, the output predictor is developed first, by minimizing the following objective function:

$$E_1 \equiv \sum_{t=0}^{NP} E_1(t) \equiv \sum_{t=0}^{NP} \sum_{k=1}^{\ell} [\hat{y}_{NN,k}(t|t-1) - y_{\text{target},k}(t)]^2 \quad (14)$$

where all variables are as previously defined. The error function to be minimized for the state predictor network is given by

$$E_2 \equiv \sum_{t=0}^{NP} E_2(t) \equiv \sum_{t=0}^{NP} \sum_{k=1}^{\ell} [\hat{x}_{NN,k}(t|t-1) - x_{\text{target},k}(t)]^2 \quad (15)$$

whereas, the error function to be minimized for the state update network is given by

$$E_3 \equiv \sum_{t=0}^{NP} E_3(t) \equiv \sum_{t=0}^{NP} \sum_{k=1}^{\ell} [\hat{x}_{NN,k}(t|t) - x_{\text{target},k}(t)]^2 \quad (16)$$

which is identical to (6). The error gradients for an FMLP and/or RMLP network trained with TF can be obtained by using the

chain-rule. The detailed computation of these gradients can be found in many recurrent network references, such as [33].

In the second phase of the off-line training process, training is performed using GF. The error functions to be minimized for the state predictor, output predictor, and state update network are similar to (14)–(16), respectively. However, in this phase the multistep state update error is minimized because the output predictor response is used in the state predictor and state update networks, instead of the output observations. Similarly, the error functions for the output and state predictors are such that MSP errors are minimized instead. The only observations used in this phase of the off-line training as network inputs are the current and past system inputs, $u(t)$. All other variables are generated by one of the three networks involved in the state filter. The detailed computation of the error gradients involved in the off-line training phase with GF are omitted, but they can be found in [34]. The convergence of the off-line training algorithm can be treated in the same manner as the nonadaptive filter case.

C. On-Line Training

Neural networks are “semiparametric” empirical models and therefore they can be “retrained” recursively as new observa-

tions become available. On-line learning of state filter networks, whether FMLP or RMLP, is complicated by the fact that the only measurements available are those of the system inputs and outputs, $\mathbf{u}(t)$ and $\mathbf{y}(t)$, respectively. Therefore, in order to correct for deviations in the filtered state value, $\hat{\mathbf{x}}(t|t)$, the updates for the weights and biases must be determined from the output residual term, $\epsilon(t)$.

As in the first phase of off-line training, during on-line training the networks are decoupled and treated separately. For all networks, only a single-step-ahead gradient propagation training is considered because on-line learning with GF becomes exceedingly impractical. As witnessed by the results presented, this approximation has minimal impact on the performance of the on-line filtering algorithm. During on-line training, the error function to be minimized is of the form

$$E(t+1) \equiv \sum_{k=1}^n [\hat{y}_{NN,k}(t+1|t) - y_{\text{target},k}(t+1)]^2 \quad (17)$$

where all variables are as previously defined. The error gradients used in off-line training must be modified for use in on-line training. For the output predictor network the error gradients can be expressed as follows:

$$\frac{\partial E(t+1)}{\partial \mathbf{w}_{\mathbf{h}_{NN}}} = 2 [\hat{\mathbf{y}}_{NN}(t+1|t) - \mathbf{y}_{\text{target}}(t+1)]^T \frac{\partial \mathbf{h}_{NN}}{\partial \mathbf{w}_{\mathbf{h}_{NN}}} \quad (18)$$

where $\mathbf{w}_{\mathbf{h}_{NN}}$ is defined as the vector containing the weights and biases of the output predictor network. For the state predictor network, the equivalent on-line training gradient can be expressed as follows:

$$\begin{aligned} \frac{\partial E(t+1)}{\partial \mathbf{w}_{\mathbf{f}_{NN}}} &= 2 [\hat{\mathbf{y}}_{NN}(t+1|t) - \mathbf{y}_{\text{target}}(t+1)]^T \\ &\times \left(\frac{\partial \mathbf{h}_{NN}}{\partial \hat{\mathbf{x}}_{NN}(t|t)} \right) \left(\frac{\partial \mathcal{K}_{NN}}{\partial \hat{\mathbf{x}}_{NN}(t|t-1)} \right) \left(\frac{\partial \mathbf{f}_{NN}}{\partial \mathbf{w}_{\mathbf{f}_{NN}}} \right) \end{aligned} \quad (19)$$

where $\mathbf{w}_{\mathbf{f}_{NN}}$ is defined as the vector containing the weights and biases of the state predictor network. Finally, for the state update network the error gradient can be expressed as

$$\begin{aligned} \frac{\partial E(t+1)}{\partial \mathbf{w}_{\mathcal{K}_{NN}}} &= 2 [\hat{\mathbf{y}}_{NN}(t+1|t) - \mathbf{y}_{\text{target}}(t+1)]^T \\ &\times \left(\frac{\partial \mathbf{h}_{NN}}{\partial \hat{\mathbf{x}}_{NN}(t|t)} \right) \left(\frac{\partial \mathcal{K}_{NN}}{\partial \mathbf{w}_{\mathcal{K}_{NN}}} \right) \end{aligned} \quad (20)$$

where $\mathbf{w}_{\mathcal{K}_{NN}}$ is defined as the vector containing the weights and biases of the state update network.

Some of the gradients contained in (18)–(20) can be computed using a sensitivity-type network once the specific network architecture (either FMLP or RMLP) is selected. The other gradients are determined in the same manner as in the nonadaptive on-line training method.

V. SIMULATION RESULTS

A. State Filter Performance Evaluation

In this study, the performance of the developed filters is assessed using the following errors:

$$\epsilon_{\text{act}}(t) = \frac{\mathbf{x}(t) - \hat{\mathbf{x}}(t|t)}{\mathbf{x}(t)}, \quad \epsilon_{\text{mod}}(t) = \frac{\mathbf{x}_m(t) - \hat{\mathbf{x}}(t|t)}{\mathbf{x}_m(t)}. \quad (21)$$

The relative error $\epsilon_{\text{act}}(t)$ is a measure of the accuracy of the state estimate $\hat{\mathbf{x}}(t|t)$, with respect to the “actual” state value $\mathbf{x}(t)$. The relative error $\epsilon_{\text{mod}}(t)$ is a measure of the accuracy of the state estimate $\hat{\mathbf{x}}(t|t)$, with respect to the state value, $\mathbf{x}_m(t)$, as obtained by any available system model, off-line signal processing procedures or infrequent measurements. The set of target state values $\mathbf{x}_{\text{target}}(t)$ and $\mathbf{x}_m(t)$ are sampled from the same process.

In the simulation results presented the quantities $\overline{\epsilon_{\text{act}}(t)}$ and $\overline{\epsilon_{\text{mod}}(t)}$, denoting the average values of the normalized residuals, $\epsilon_{\text{act}}(t)$ and $\epsilon_{\text{mod}}(t)$, respectively, have also been calculated. Furthermore, the normalized mean-square error with respect to the “actual” state value (E_{NMSE}) is used as a performance measure. It is defined as the mean-squared deviation of the filtered state with respect to the actual states over NP samples, normalized by the mean squared value of the actual state for the same number of samples.

B. An Artificial Nonlinear System

1) *Problem Description*: The 2I2O stochastic nonlinear system used in this example is assumed to be described by the following state and output equations, shown in (22) at the bottom of the page, and

$$\begin{cases} y_1(t+1) = 0.7(x_1(t+1) + x_2(t+1)) \\ y_2(t+1) = 1.5x_1^2(t+1) \end{cases} \quad (23)$$

respectively, where

$$\begin{array}{ll} x_1(t), x_2(t), \text{ and } x_3(t) & \text{states of the system;} \\ u_1(t) \text{ and } u_2(t) & \text{two inputs;} \\ y_1(t) \text{ and } y_2(t) & \text{two outputs of the system.} \end{array}$$

The variable $n_{x_i}^{\text{process}}(t)$ is a zero-mean white Gaussian process noise added to the system. The system parameters α , β and γ take on different values as follows:

$$\begin{aligned} \alpha &= \begin{cases} 0.50, & \text{Actual System} \\ 0.50, & \text{Model 1} \\ 0.45, & \text{Model 2} \end{cases} \\ \beta &= \begin{cases} \frac{2}{3}, & \text{Actual System} \\ \frac{1}{3}, & \text{Model 1} \\ \frac{3}{5}, & \text{Model 2} \end{cases} \\ \gamma &= \begin{cases} 0.30, & \text{Actual System} \\ 0.25, & \text{Model 1} \\ 0.25, & \text{Model 2} \end{cases}. \end{aligned} \quad (24)$$

The aforementioned parameters indicate that *Model 1* is more accurate than *Model 2*, as compared to the *Actual System*.

$$\begin{cases} x_1(t+1) = \alpha(x_1(t))^\beta + 0.3x_2(t)x_3(t) + 0.2u_1(t) + n_{x_1}^{\text{process}}(t) \\ x_2(t+1) = \alpha(x_2(t))^\beta + \gamma x_3(t)x_1(t) + 0.5u_1(t) + n_{x_2}^{\text{process}}(t) \\ x_3(t+1) = \alpha(x_3(t))^\beta + \gamma x_1(t)x_2(t) + 0.5u_2(t) + n_{x_3}^{\text{process}}(t) \end{cases} \quad (22)$$

2) *Development of the Nonadaptive and Adaptive Neural State Filters:* It is assumed that $x_3(t)$ is the state to be estimated. The neural state filtering algorithms proposed are now applied to develop a nonadaptive and an adaptive filter. First, we perform off-line training, followed by on-line training.

The design of both the nonadaptive and adaptive state filters requires an estimation data set and a validation (or testing) data set. The former is divided into a training and evaluation set and it is used to specify the filter structure and parameters, whereas the latter is used in final testing of the filter. While using the validation set the filter structure and parameters are not adjusted. The estimation and validation data sets are a collection of step, ramp and sines, that cover a wide range of typical inputs to which the 2I2O system is subjected. The estimation data set is generated using *Model 1* for the nonadaptive filter and *Model 2* for the adaptive filter. In addition, process noise is added to the equations while data is collected. The process noise $n_{x_i}^{\text{process}}(t)$ is assumed to be zero-mean white Gaussian with 0.01 standard deviation (sd). The validation data set is generated using (22) and (23). An *estimation* data set is obtained that consists of 1000 data samples. The estimation data set consists of a *training set* (600 samples) and an *evaluation set* (consisting of 400 samples). The evaluation data set is used to determine the best stopping point in the training process (to prevent over-training) as well as to select the filter structure.

Only one-hidden-layer networks are used, where the network size is determined using the following approach: a given network is trained using the training data set and the training process is stopped when the error associated with the evaluation data set began to increase (indicating over-training); the error in the evaluation data set is noted and training is repeated with a different size network; a number of network sizes is examined and the architecture that resulted in the lowest evaluation data set error is chosen. For the nonadaptive case the neural network state filter is a 5-6-1 FMLP. The average error on the evaluation portion of the estimation data set for this architecture is $E_{NMSE} = 0.285\%$. For the adaptive case there are three neural networks involved in the state filter. The output predictor network was determined to be a 5-6-2 FMLP network, with a corresponding evaluation data set error of $E_{NMSE} = 0.183\%$. The state predictor network is determined to be a 5-8-1 FMLP network, with a corresponding evaluation data set error of $E_{NMSE} = 0.612\%$. Finally, the state update network is determined to be a 5-6-1 FMLP network, with a corresponding evaluation data set error of $E_{NMSE} = 0.213\%$.

3) *Neural State Filter Comparison:* In this section a comparison of various network architectures is presented, evaluating their effectiveness to address the state filtering problem at hand. Table I³ presents a summary of these results. The comparison is performed by running once the *evaluation* data set through the various networks. Both feedforward and recurrent networks are considered. The two predictor networks are dynamic in nature, whereas the state update network is static. The architectures depicted in the table represent the best network choices, selected following a search over possible network configurations and input layer layouts.

³(Architecture column) All predictor networks are dynamic, whereas all state update networks are static.

TABLE I
2I2O SYSTEM NEURAL STATE FILTER COMPARISON RESULTS

Neural State Filter		Architecture	Network Size	E_{NMSE}
Non-adaptive Filter	State Update	Feedforward	5-6-1	0.29%
	State Update	Recurrent	5-6-1	3.04%
Adaptive Filter	Output Predictor	Feedforward	5-6-2	0.18%
	Output Predictor	Recurrent	5-7-2	16.6%
	State Predictor	Feedforward	5-8-1	0.61%
	State Predictor	Recurrent	5-10-1	22.3%
Filter	State Update	Feedforward	5-6-1	0.21%
	State Update	Recurrent	5-8-1	17.5%

TABLE II
VALIDATION DATA SET INPUT SIGNALS FOR 2I2O SYSTEM

$u_1(t)$	$u_2(t)$	Number of Samples
$0.3 \sin(\pi t/8) + 0.5(t/400)$	$0.45(t/400) - 0.2 \sin(\pi t/20)$	100
$0.3, t \leq 26s$ $0, t > 26s$	$0.3, t \leq 36s$ $0, t > 36s$	50
$0.002t$	0.2	50
0.2	$0.002t$	50
$0.3 + 0.2 \sin(\pi t/5)$	$0.2 + 0.3(t/200)$	200

The comparisons of Table I clearly indicate that for this problem the feedforward architecture is preferable to the recurrent architecture. This is to be expected for the state update network as this mapping does not implement dynamic effects. A possible explanation for the predictor networks could be that this example is relatively simple and the underlying dynamics can be captured by the global feedback implemented in a feedforward network.

4) *Neural Filter Validation:* The resulting nonadaptive and adaptive filters consist of static and dynamic feedforward networks. Following complete selection of the filter architecture, the combined set of networks involved is tested by applying the *validation* data set (consisting of 450 samples which the network has not previously seen). Table II shows the validation data set used. Two experiments are considered for the validation set: a low noise experiment with noise sd 0.01 and a high noise experiment with noise sd 0.05. Fig. 3 shows the nonadaptive and adaptive state filtering results for the validation data set with high noise level. None of the neural state filters account for on-line learning. Table III^{4,5} displays the average state estimation errors, using the previously defined metrics. The errors with respect to the actual states without on-line learning are large. This is because the model used in obtaining the state values for training has been intentionally made to exhibit significant errors. Off-line training cannot reduce these errors any further. However, as shown in Table III on-line training has been successfully used to further reduce the state filtering errors with

⁴(Columns 3, 4, 5, 6) Without on-line learning.

⁵(Column 7) With on-line learning.

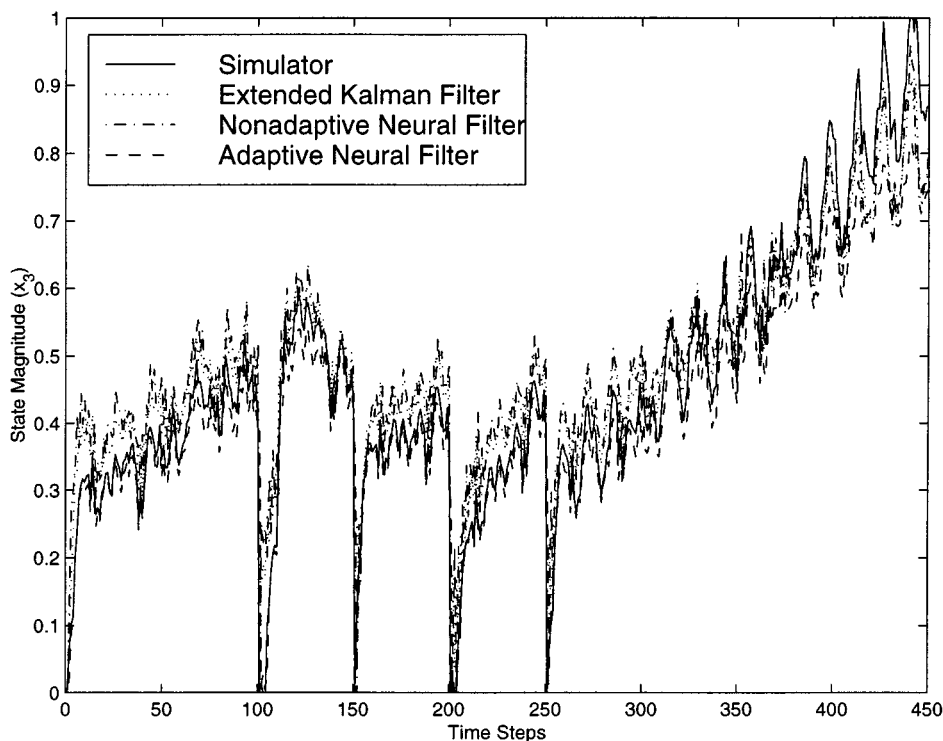


Fig. 3. State filter responses using the validation data set (high noise environment); neural state filters without on-line learning.

TABLE III
2120 SYSTEM $x_3(t)$ STATE FILTER VALIDATION SET ERRORS

State Filter		E_{NMSE}	$\overline{\epsilon_{act}(t)^2}$	$\overline{\epsilon_{mod}(t)^5}$	$\overline{\epsilon_{act}(t)^3}$	$\overline{\epsilon_{mod}(t)}$
Non-adaptive Neural Filter	Low Noise ($\sigma = 0.01$)	0.32%	12.5%	0.1%	8.5%	0.2%
	High Noise ($\sigma = 0.05$)	0.46%	11%	0.05%	9.5%	0.15%
Adaptive Neural Filter	Low Noise ($\sigma = 0.01$)	0.32%	10%	0.1%	0.8%	2.3%
	High Noise ($\sigma = 0.05$)	0.26%	11%	0.05%	0.9%	2.5%
Extended Kalman Filter	Low Noise ($\sigma = 0.01$)	0.30%	N.A.	N.A.	12%	0.05%
	High Noise ($\sigma = 0.05$)	0.45%	N.A.	N.A.	11%	0.03%

respect to the actual states. Due to space limitations the estimation results with on-line learning are only tabulated. The significant drop in the adaptive filter state estimation error can be clearly observed.

5) *EKF*: For comparison purposes, an EKF is designed for estimating the state $x_3(t)$. The standard EKF algorithm is used in developing a nonadaptive filter. For a fair comparison with the nonadaptive neural state filter, *Model 1* and its linearized version is used in the EKF design. Specifically, in the *Prediction Step*, *Model 1* is used, whereas the same model is linearized for use in the covariance matrix calculations. In the *Update Step* of the EKF, the linearized version of *Model 1* is used in both the update equation of the covariance matrix and the EKF gain matrix calculations. Finally, the noise covariance matrices are calculated using the white Gaussian noise assumptions. The EKF gains are recomputed following the arrival of each new observation. As a result, the performance of an EKF should be compared to that of a nonadaptive neural filter with on-line learning.

An attempt was made to develop a linearized KF. The results, however, were not satisfactory because in most simulation studies the filter did not converge. The comparative results of the designed EKF are presented in Table III and Fig. 3. In this figure, three state filters are compared with the “actual” state values of the system. The neural state filters shown do not include the effects of on-line learning. Due to space limitations we have included the effects of on-line learning in the next case study. The EKF exhibits response similar to both the nonadaptive and adaptive neural filters without on-line learning. During the initial segment of the simulations shown in Fig. 3, the nonadaptive neural filter and the EKF exhibit worse performance than the adaptive neural filter. However, in the latter segments of this simulation study the EKF appears to perform better than both the nonadaptive and adaptive neural filters, again without the on-line learning effects. Overall, the average responses of the three filters are comparable when no on-line learning is used. The numerical results of the state estimation error with on-line learning are included in Table III. A quick review of the results

with on-line learning reveals a shift in the state estimation errors. The nonadaptive estimation error with on-line learning is improved with respect to the actual states. The error with respect to the model is worse. The same trend is observed for the adaptive estimation error with on-line learning. Despite a significant drop in the estimation error with respect to the actual states, the error with respect to the model further increases. This can be explained on the basis that on-line learning is expected to give better state estimates with respect to the actual states and further the model states exhibit inaccuracies with respect to the actual states.

C. A Complex Thermal-Hydraulic Process System

In this section the adaptive neural state filtering algorithm is applied to the UTSG and compared with EKF designs. The UTSG is a highly nonlinear, unstable, multivariable thermal-hydraulic process system used in nuclear power plants with pressurized water reactors (PWRs). The UTSG is used to convert water in liquid phase to steam. The steam from the UTSG is used to drive turbines which in turn drive electric generators. The UTSG is, therefore, an important power plant component whose operation is critical for the overall plant operation. The UTSG is sufficiently complex that it is considered a real-world test case for the proposed algorithms. In this case study, two adaptive neural state filters are developed for filtering slowly varying process parameters; the riser void fraction $\alpha_R(\tau)$ and the overall primary heat transfer coefficient U_{over} .

1) *Description of the UTSG Dynamics:* In UTSG operation, high pressure liquid from the power plant primary loop flows in and out of primary side (also called the 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. Feedwater enters the UTSG 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 riser and more separation occurs near the top of the UTSG. 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.

The three outputs of a UTSG that are usually measured are the cold-leg temperature $T_{cl}(t)$, the downcomer water level $L_w(t)$, and the secondary steam pressure $P_s(t)$; while the five disturbances acting upon the system are the feedwater temperature $T_{fw}(t)$, the hot-leg temperature $T_{hl}(t)$, the primary mass flow rate $W_{pr}(t)$, the primary pressure $P_{pr}(t)$, and the steam flow rate $W_{st}(t)$. There is only one manipulated control input which is the feedwater flow rate $W_{fw}(t)$.

The open-loop dynamics of a UTSG are unstable throughout its operating regime. Furthermore, during operation the water level must always be maintained between certain prespecified limits to 1) make sure that the steam quality is always above a certain value to prevent turbine blade damage; 2) have effective

heat transfer from primary side to the secondary side; and 3) prevent hydro-dynamic instabilities in the feed-water pipes due to water hammers. Because the open-loop system is unstable, a stabilizing controller is required to allow system operation. The controller structure used in this study is that proposed by Menon and Parlos [27].

2) *UTSG Simulator:* A UTSG simulator developed by Strohmayer [41] and modified by Choi [8], is adopted for this study. The 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. There are seven control volumes in the model spatial domain: four on the secondary side, the steam dome-downcomer (SDD) region, the tube bundle region and the riser region, which is divided into a saturated volume and a subcooled volume. The three control volumes on the primary side are the inlet plenum, the outlet plenum, and the fluid volume within the tube bundle region.

A brief description of the UTSG simulator used is presented here, for the reader to appreciate the complexity of the involved process dynamics. For the primary side model, a set of three differential equations with three unknowns is used. In matrix form these are

$$\mathbf{E}_1(\mathbf{T}(\tau)) \dot{\mathbf{T}}(\tau) = \mathbf{f}_p(\mathbf{T}(\tau), Q_B(\tau), T_{hl}(\tau), P_{pr}(\tau), W_{pr}(\tau)) \quad (25)$$

where we have

$$\mathbf{E}_1(\mathbf{T}(\tau)) = \begin{bmatrix} E_{11}(\mathbf{T}(\tau)) & 0 & 0 \\ 0 & E_{22}(\mathbf{T}(\tau)) & 0 \\ 0 & 0 & E_{33}(\mathbf{T}(\tau)) \end{bmatrix} \quad (26)$$

$$\mathbf{T}(\tau) = [T_1(\tau), T_2(\tau), T_3(\tau)]^T \quad (27)$$

where

- τ continuous time;
- $\mathbf{f}_p(\cdot)$ three-dimensional vector forcing function;
- $T_1(\tau)$ temperatures of the inlet;
- $T_2(\tau)$ temperature of the tube bundle;
- $T_3(\tau)$ temperature of the outlet;
- $\mathbf{E}_1(\cdot)$ diagonal matrix function of the temperature vector;
- $\mathbf{T}(\tau)$ dependence on the temperature vector is complicated by the embedded empirical correlations governing the thermal and fluid properties of the system.

The thermal load $Q_B(\tau)$ is the thermal energy transferred from the primary side to the secondary side across the tube bundle region. The heat load is calculated using

$$Q_B(\tau) = U_{\text{over}} A_o \Delta T_{LM}(\tau) \quad (28)$$

where U_{over} is the overall heat transfer coefficient, A_o is the total outside surface area of the tubes, and $\Delta T_{LM}(\tau)$ is the log-mean temperature difference given by

$$\Delta T_{LM}(\tau) = \frac{T_1(\tau) - T_2(\tau)}{\ln \left[\frac{T_1(\tau) - T_{\text{sat}}}{T_2(\tau) - T_{\text{sat}}} \right]} \quad (29)$$

where T_{sat} is the saturation temperature of water at pressure P_{pr} .

For the secondary side, the mass and energy conservation equations are summed up over the control volumes and the mo-

mentum equation is used to describe the recirculation flow. The secondary equations can then be represented as

$$\mathbf{E}_2(\mathbf{x}_s(\tau)) \dot{\mathbf{x}}_s(\tau) = \mathbf{f}_s(\mathbf{x}_s(\tau), Q_B(\tau), W_{fw}(\tau), W_{st}(\tau), T_{fw}(\tau), \mathbf{v}(\tau)) \quad (30)$$

or

$$\mathbf{E}_2(\mathbf{x}_s(\tau)) \frac{d}{d\tau} \begin{bmatrix} U_o(\tau) \\ V_v(\tau) \\ \alpha_N(\tau) \\ \alpha_R(\tau) \\ P_s(\tau) \\ \bar{W}(\tau) \end{bmatrix} = \begin{bmatrix} f_1(\tau) + Q_B(\tau) \\ f_2(\tau) \\ f_3(\tau) \\ f_4(\tau) \\ f_5(\tau) \\ f_6(\tau) \end{bmatrix} \quad (31)$$

where

- $\mathbf{x}_s(\tau)$ secondary states;
- $U_o(\tau)$ internal energy at the downcomer exit;
- $V_v(\tau)$ vapor volume in the steam dome;
- $\alpha_N(\tau)$ void fractions at the riser inlet;
- $\alpha_R(\tau)$ void fractions at the riser outlet;
- $P_s(\tau)$ secondary side steam pressure;
- $\bar{W}(\tau)$ recirculation flow rate.

The process noise is modeled by $\mathbf{v}(\tau)$. The right-hand-side of (30), that is $\mathbf{f}_s(\cdot)$, represents the forcing functions and is coupled to the primary side through the thermal load $Q_B(\tau)$. The scalar functions in (31), $f_i(\tau)$ for $i = 1, \dots, 6$, represent very complex functions of the states derived from a combination of physical principles and empirical correlations. Each of these expressions is several pages long. The nonlinear matrix function $\mathbf{E}_2(\cdot)$ is dependent upon the state vector $\mathbf{x}_s(\tau)$ also and it is characterized by similar complexity as the matrix $\mathbf{E}_1(\cdot)$.

The system of (25)–(31) are combined to obtain the following nonlinear state-space representation for the UTSG:

$$\mathbf{E}(\mathbf{x}(\tau)) \dot{\mathbf{x}}(\tau) = \mathbf{f}_{ps}(\mathbf{x}(\tau), u(\tau), \mathbf{w}(\tau)) \quad (32)$$

where

- $\mathbf{E}(\cdot)$ matrix dependent on $\mathbf{E}_1(\cdot)$ and $\mathbf{E}_2(\cdot)$;
- $\mathbf{x}(\tau)$ combined primary and secondary side state vector;
- $u(\tau)$ controlled input, $W_{fw}(\tau)$;
- $\mathbf{w}(\tau)$ disturbance vector;
- $\mathbf{f}_{ps}(\cdot)$ nonlinear vector function, representing the UTSG dynamics.

The system of these nine nonlinear ordinary differential equations is solved numerically to advance the transient simulation, given values for $u(\tau)$ and $\mathbf{w}(\tau)$ and an initial value for the state $\mathbf{x}(\tau)$. The measured UTSG outputs are expressed in terms of the system states as follows:

$$\mathbf{y}(\tau) = \mathbf{h}_{ps}(\mathbf{x}(\tau), \mathbf{w}(\tau)) \quad (33)$$

where

$$\mathbf{y}(\tau) = [L_w(\tau), T_3(\tau), P_s(\tau)]^T \quad (34)$$

and where $\mathbf{h}_{ps}(\cdot)$ is a three-dimensional nonlinear function vector and $T_3(\tau)$ is assumed equal to the cold-leg temperature, $T_{cl}(\tau)$. Fig. 4 shows a block diagram of the UTSG simulator used in this study, with the input, disturbances, and outputs denoted in discrete-time form.

It should be pointed out that (32) and (33) represent a UTSG simulator. Nevertheless, this simulator cannot be used directly

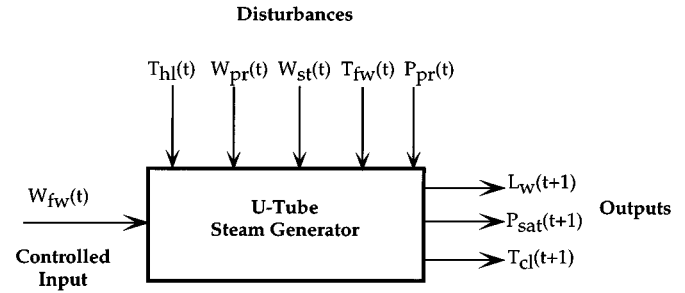


Fig. 4. Block diagram of the UTSG process simulator.

in state filtering methods, such as the EKF, in the form of a standard state-space model. This is because the nonlinear forms involved in the simulator are complex expressions incorporating empirical relations not suitable for analytic differentiation.

3) *Development of the Neural State Filters:* In this section, two adaptive neural state filters are developed and investigated. The first state filter is developed to estimate the riser void fraction, $\alpha_R(t)$. The second state filter is developed to estimate the primary heat transfer coefficient (HTC), U_{over} , which is a constant parameter.

The estimation and validation data sets used in the development and evaluation of the neural models are a collection of UTSG process simulator data corresponding to step and ramp changes in operating power level. These operating power level changes are effected by changes in the UTSG inputs: the hot-leg temperature $T_{hl}(t)$, the steam flow rate $W_{st}(t)$, the feedwater temperature $T_{fw}(t)$, and the controlled input, the feedwater flow rate $W_{fw}(t)$. These disturbances and the controlled input will be denoted collectively, by the vector $\mathbf{u}(t)$. In the riser void fraction adaptive filter, the only output measurement used is the UTSG water level $L_w(t)$. In the HTC adaptive filter, all three output measurements of the UTSG process simulator are used. In addition, process and measurement noise, which is zero-mean, white, Gaussian with 0.05 standard deviation (sd), is added to the collected data.

The detailed estimation and validation data sets used in the design and validation of the adaptive neural filters are given in [28]. All simulations of the UTSG are performed in the low power operation range which corresponds to less than 20% of full operational power. This is because the low power operation regime reveals the greatest nonlinearities and nonminimum phase behavior in the dynamics of the UTSG. The following sections briefly discuss the design and evaluation of the neural state filters as applied to the UTSG.

a) *Adaptive Riser Void Fraction Filter:* The adaptive state filter algorithm is used to estimate the riser void fraction, $\alpha_R(t)$. The estimation data set used in training the networks comprises of data collected from the UTSG subjected to different step and ramp changes in the operating power level. The state $\alpha_{target,R}(t)$ is obtained from the UTSG simulator and is used to develop the appropriate neural models.

Three neural networks are used in this estimation scheme: the output predictor, the state predictor, and the state update network. The development of these networks follows the earlier discussion on the algorithms. The neural filter setup used in this case study is shown in Fig. 5.

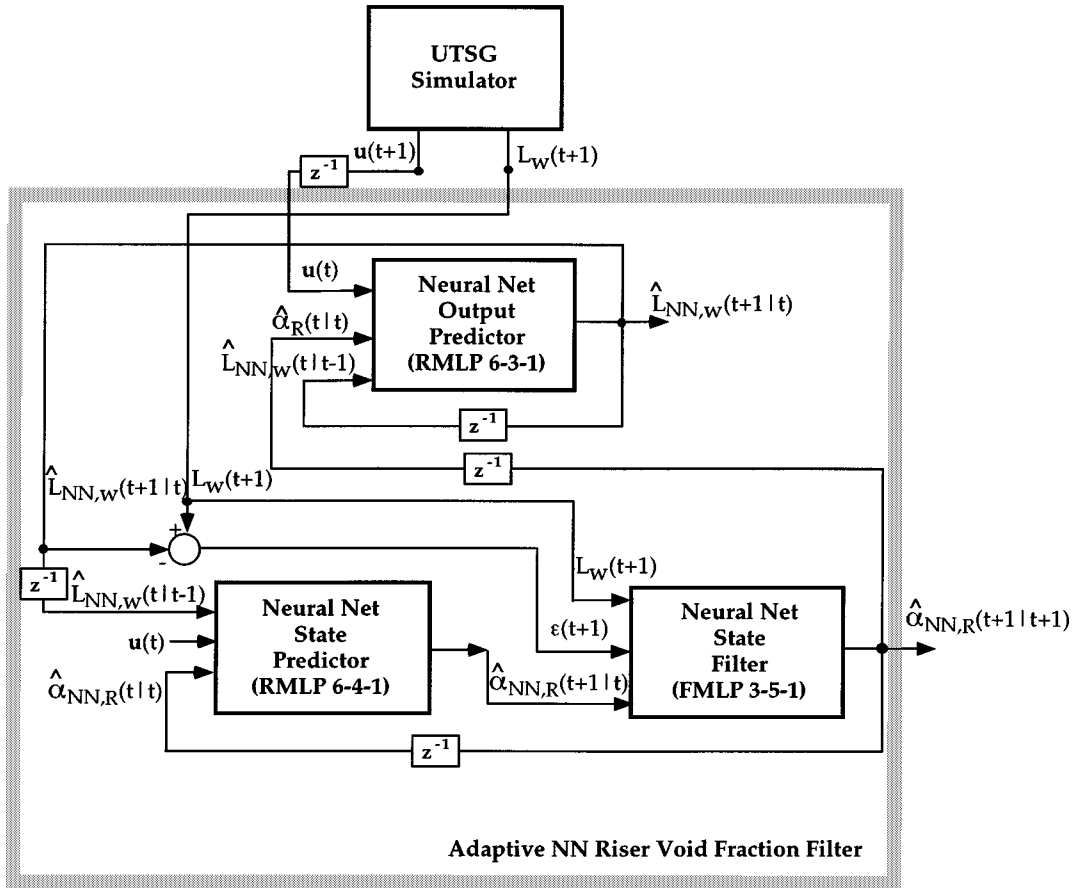


Fig. 5. Block diagram of the UTSG riser void fraction adaptive filter.

b) Neural Output Predictor: The neural output predictor architecture is initially chosen to be a three-layer RMLP network with six nodes in the first layer, corresponding to six inputs and one node in the third layer, corresponding to one output. The single output of the third layer is the single-step-ahead prediction (SSP) of the UTSG water level, $\hat{L}_{NN,w}(t+1|t)$. The six inputs to the first layer are the process inputs, $\mathbf{u}(t)$, the state $\alpha_{\text{target},R}(t)$ and the delayed output of the neural model itself, $\hat{L}_{NN,w}(t|t-1)$. Initially, the number of nodes in the hidden layer is set to three. The neural output predictor is then trained with the estimation data set. The target is the UTSG simulator output, $L_{\text{target},w}(t+1)$.

The training of the neural output predictor is continued until the stopping criterion is reached; that is, until the E_{NMSE} of the evaluation data set (no weight update) just begins to increase. The number of nodes in the hidden layer is increased and the training cycle is repeated. A suitable number of neural networks are developed in this way and the best among them is chosen. In this case, the best neural output predictor architecture is found to be 6-3-1. The E_{NMSE} of the output, in this case, is 0.4%.

c) Neural State Predictor: The neural output predictor weights are now fixed and no further training is performed on this model. The neural state predictor is chosen to be a three-layer RMLP with six inputs in the first layer, corresponding to six inputs, and one node in the third layer, corresponding to one output. The output of the third layer is the SSP of the state $\hat{\alpha}_{NN,R}(t+1|t)$. The inputs to the first layer are

the output from the neural output predictor $\hat{L}_{NN,w}(t|t-1)$, the inputs to the process system $\mathbf{u}(t)$, and the delayed output of the neural state predictor $\hat{\alpha}_{NN,R}(t|t-1)$. The number of nodes in the hidden layer is set to three initially. The state predictor is then trained with the estimation data set. The target is the state $\alpha_{\text{target},R}(t+1)$ as provided by the UTSG simulator.

The neural state predictor is trained using the approach summarized for the output predictor. Based on the lowest E_{NMSE} on the evaluation data set (no weight update), a RMLP network with an architecture of 6-4-1 is determined to be the best. The average E_{NMSE} for this architecture is 0.1%.

d) Neural State Update: The weights of the neural output and state predictors are fixed and no further training is performed on these networks. The neural state update architecture is first chosen to be a three-layer FMLP networks with three nodes in the input (first) layer and one node in the third layer, corresponding to one output. The output of the third layer is the filtered state value, $\hat{\alpha}_{NN,R}(t+1|t+1)$. The three inputs to the first layer are the UTSG process simulator outputs $L_{\text{target},w}(t+1)$, the residual $\epsilon(t+1)$, where $\epsilon(t+1) = L_{\text{target},w}(t+1) - \hat{L}_{NN,w}(t+1|t)$, and the output from the neural state predictor $\hat{\alpha}_{NN,R}(t+1|t)$. The target in training is the same as in the training of the neural state predictor, namely the state $\alpha_{\text{target},R}(t+1)$ obtained from the UTSG simulator.

The initial number of nodes in the hidden layer of the neural state filter is set to four and training is performed as described

in the previous section. The best FMLP network is found to be a 3-5-1 network with the average E_{NMSE} computed as 0.1%.

e) Adaptive Heat Transfer Coefficient Filter: The adaptive filtering algorithm is also used to estimate the primary heat transfer coefficient (HTC), U_{over} . The qualitative difference between U_{over} and $\alpha_R(t)$ is that U_{over} is a very slowly changing parameter of the UTSG and it can be considered to have a *constant* value for the time duration of normal operating transients. The HTC can still be considered a “state” with no dynamics for applying the adaptive state filtering algorithm.

It is apparent that the estimation data set used to develop the neural filter should cover a wide range of transients of the UTSG process system with *different* U_{over} values. In addition, these variations in U_{over} should be *observable* in the UTSG output measurements. Based on trial runs, it was found that changes in U_{over} are *weakly* observable through the UTSG simulator outputs, $L_w(t)$, $P_s(t)$ and $T_{cl}(t)$. Therefore, in order to infer variations in U_{over} with high accuracy, all three UTSG simulator outputs were used in the filter. Further, it was found that the steam flow rate, $W_{st}(t)$, greatly obscured the “observability” of U_{over} through the UTSG outputs. For this reason, the estimation and validation data set are designed with the steam flow rate, $W_{st}(t)$, kept constant during power level changes. This is clearly unrealistic; however, the intention in this study is to determine the feasibility of determining the “constant” parameter U_{over} using an adaptive neural filter.

Three neural models are used in this filtering scheme: the neural output predictor, the neural state predictor and the neural state update network. The development of these models, again, follows the relevant earlier discussion on the filtering algorithms. The filter setup for this case study is shown in the block diagram in Fig. 5.

f) Neural Output Predictor: The neural output predictor consists of three networks; a single network is developed for each of the outputs of the UTSG process simulator, $L_w(t)$, $T_{cl}(t)$, and $P_s(t)$, denoted by the vector $\mathbf{y}(t)$. Each of the networks in the output predictor is a three-layer RMLP network. The same eight inputs are used for each of the networks. These are the four inputs of the process system: $\mathbf{u}(t)$, the primary heat transfer coefficient U_{over} , and the three neural output predictor outputs (delayed by one time step), $\hat{\mathbf{y}}_{NN}(t|t-1)$. The output for each of the networks is $\hat{\mathbf{y}}_{NN}(t+1|t) = [\hat{L}_{NN,w}(t+1|t), \hat{T}_{NN,cl}(t+1|t) \text{ and } \hat{P}_{NN,s}(t+1|t)]$. The number of hidden nodes in each of the networks is initially set to four. The three neural models are then trained concurrently with the estimation data set. The targets for each of the networks are $L_{\text{target},w}(t+1)$, $T_{\text{target},cl}(t+1)$ and $P_{\text{target},s}(t+1)$, respectively, and are obtained from the UTSG simulator. Training is stopped when the E_{NMSE} of the network outputs for the evaluation data set (no weight update) just begins to increase. In this case, the best neural output predictor is found to be 8-4-1 for $L_w(t)$ and $T_{cl}(t)$ predictors and 8-5-1 for the $P_s(t)$ predictor. The E_{NMSE} s for the three networks modeling the three process outputs are 0.7% for the $L_w(t)$ predictor, 0.00004% for the $T_{cl}(t)$ predictor and 0.002% for the $P_s(t)$ predictor. These E_{NMSE} values are computed using the evaluation data set.

g) Neural State Predictor: The neural output predictor weights are now fixed and no further training is performed

on this network. The neural state predictor is chosen to be a three-layer FMLP with eight inputs in the first layer, corresponding to eight inputs, and one node in the third layer, corresponding to one output. The output of the third layer is the SSP of the HTC, $\hat{U}_{NN,\text{over}}(t+1|t)$. The inputs to the first layer are the outputs from the neural output predictor, $\hat{\mathbf{y}}_{NN}(t|t-1)$, the process system inputs, $\mathbf{u}(t)$ and the delayed output of the neural state predictor, $\hat{U}_{NN,\text{over}}(t|t-1)$. The number of nodes in the hidden layer is set to three initially. The state predictor is then trained with the estimation data set. The target is the HTC, $U_{\text{target},\text{over}}$, values which are obtained from the UTSG simulator.

The neural state predictor is trained until the E_{NMSE} of the evaluation data set (no weight update) just begins to increase, as reported for the output predictors. The FMLP network with the lowest error is found to be a 8-5-1 network and the average E_{NMSE} for the evaluation data set is computed at 0.1%.

h) Neural State Update: The weights of the neural output and state predictors are fixed and no further training is performed on these networks. The neural state update architecture is first chosen to be a three-layer FMLP network with seven nodes in the input (first) layer, corresponding to seven inputs, and one node in the third layer, corresponding to one output. The output of the third layer is $\hat{U}_{NN,\text{over}}(t+1|t+1)$. The seven inputs to the first layer are the UTSG simulator outputs, $\mathbf{y}(t+1)$, the residuals, $\epsilon(t+1)$ where $\epsilon(t+1) = \mathbf{y}(t+1) - \hat{\mathbf{y}}_{NN}(t+1|t)$ and the output from the state predictor, $\hat{U}_{NN,\text{over}}(t+1|t)$. The target in training is the same as in the training of the state predictor, namely, the HTC, $U_{\text{target},\text{over}}$ obtained from the UTSG simulator.

The initial number of nodes in the hidden layer is set to four and training is performed as described in the previous section. The best FMLP network is found to be a 7-4-1 network with the average E_{NMSE} in the evaluation data set as 0.1%.

4) Neural State Filter Comparison: A comparison of various network architectures is now presented, demonstrating their effectiveness to address the state filtering problems of the complex process system considered. Table IV presents a summary of these results. The comparison is performed by running once the *evaluation* data set through the various networks. Both feedforward and recurrent networks are considered. The two predictor networks are dynamic, whereas the state update network is static. The architectures depicted in the table represent the best network choices, selected following a search over possible network configurations and input layer layouts.

There are a couple of observations to be made about Table IV. The comparison clearly indicates that for this problem, the recurrent architecture is preferable to the feedforward architecture. The exception is the state update network. Again, this is expected for the state update network because this mapping does not capture dynamic effects. Furthermore, for the riser void fraction adaptive filter the state update network results using feedforward and recurrent networks are comparable. The table supports the assertion that as the complexity of the dynamics to be modeled increases, dynamic recurrent networks might become preferable to dynamic feedforward networks for predictive purposes. This assertion is consistent with our experience in developing MSP for processes with varying level of complexity [34].

TABLE IV
UTSG NEURAL STATE FILTER COMPARISON RESULTS

Neural State Filter		Architecture	Network Size	E_{NMSE}
Adaptive Riser Void	Output Predictor	Feedforward	6-4-1	16.2%
	Output Predictor	Recurrent	6-3-1	0.4%
	State Predictor	Feedforward	6-4-1	12.5%
	State Predictor	Recurrent	6-4-1	0.14%
Fraction Filter	State Update	Feedforward	3-5-1	0.13%
	State Update	Recurrent	3-4-1	0.2%
Adaptive Heat Trasfer	Output Predictor 1	Feedforward	8-5-1	8.2%
	Output Predictor 1	Recurrent	8-4-1	0.67%
	Output Predictor 2	Feedforward	8-5-1	6.8%
	Output Predictor 2	Recurrent	8-4-1	0.00004%
	Output Predictor 3	Feedforward	8-5-1	7.3%
	Output Predictor 3	Recurrent	8-5-1	0.002%
Coefficient Filter	State Predictor	Feedforward	8-7-1	12%
	State Predictor	Recurrent	8-5-1	0.13%
	State Update	Feedforward	7-4-1	0.11%
	State Update	Recurrent	7-3-1	13.2%

5) *Validation of the Neural State Filters:* The adaptive filters for the riser void fraction and HTC are now tested using the validation (or previously unseen) data set. The selected adaptive filters consist of recurrent predictors and feedforward state update mapping. The validation set consists of two individual tests; a ramp increase from 5% to 10% with 0.6%/min and a step increase from 15% to 18% in the UTSG simulator power level. Zero-mean white Gaussian process and measurement noise with 0.05 sd is initially added to the validation data sets. The performance of the filters is later evaluated with a higher level of process and measurement noise; the validation set was also corrupted with zero-mean white Gaussian with 0.15 sd.

The validation data set used in evaluating the HTC adaptive filter is listed in Table V. There are eight individual tests in the validation data set; Tests 1 to 4 are conducted at steady-state power levels, Tests 5 and 7 are ramp increases in power and Tests 6 and 8 are step changes in power. These last four tests are conducted with different U_{OVER} values and with zero-mean white Gaussian process and measurement noise, with 0.05 sd, added to the validation data set. The performance of the filters to Test 5 and 6 is then evaluated with a higher level of process and measurement noise. The noise level is set at 0.15 sd.

Riser Void Fraction Filter: The performance of the riser void fraction adaptive filter on the validation data set corrupted with both low and high levels of noise is evaluated. Two filters are compared, without and with on-line learning. The average E_{NMSE} for the riser void fraction filtered value, $\hat{\alpha}_R(t|T)$ and the average errors $\overline{\epsilon_{act}(t)}$ and $\overline{\epsilon_{mod}(t)}$ values are listed in Table VI. The riser void fraction estimated values for the ramp input in the validation data set are shown in Fig. 7, the estimated values for the step input in the validation data set are shown in Fig. 8.

TABLE V
VALIDATION DATA SET USED IN THE UTSG HTC ADAPTIVE FILTER

U_{over} (% of Nominal Value)	Load Changes (% of Full Power)
70%	5% (Steady-State)
60%	10% (Steady-State)
50%	15% (Steady-State)
70%	20% (Steady-State)
65%	15% to 20% (Ramp) (1%/minute)
45%	15% to 18% (Step)
55%	15% to 20% (Ramp) (1%/minute)
35%	10% to 13% (Step)

The state filtering results obtained in the validation tests indicate that both the riser void fraction adaptive filters perform quite well. The adaptive filter performance is further enhanced by allowing on-line learning. The average state estimation error with respect to the actual states is reduced from a maximum value of 2.6% to about 0.65%. At the same time, the estimation error with respect to the model increases as a result of the on-line learning. This is to be expected as discussed in the previous case study. Considering the complexity of the underlying process, the results from both filters are quite encouraging. In fact, for most of the simulations performed using the neural filter with on-line

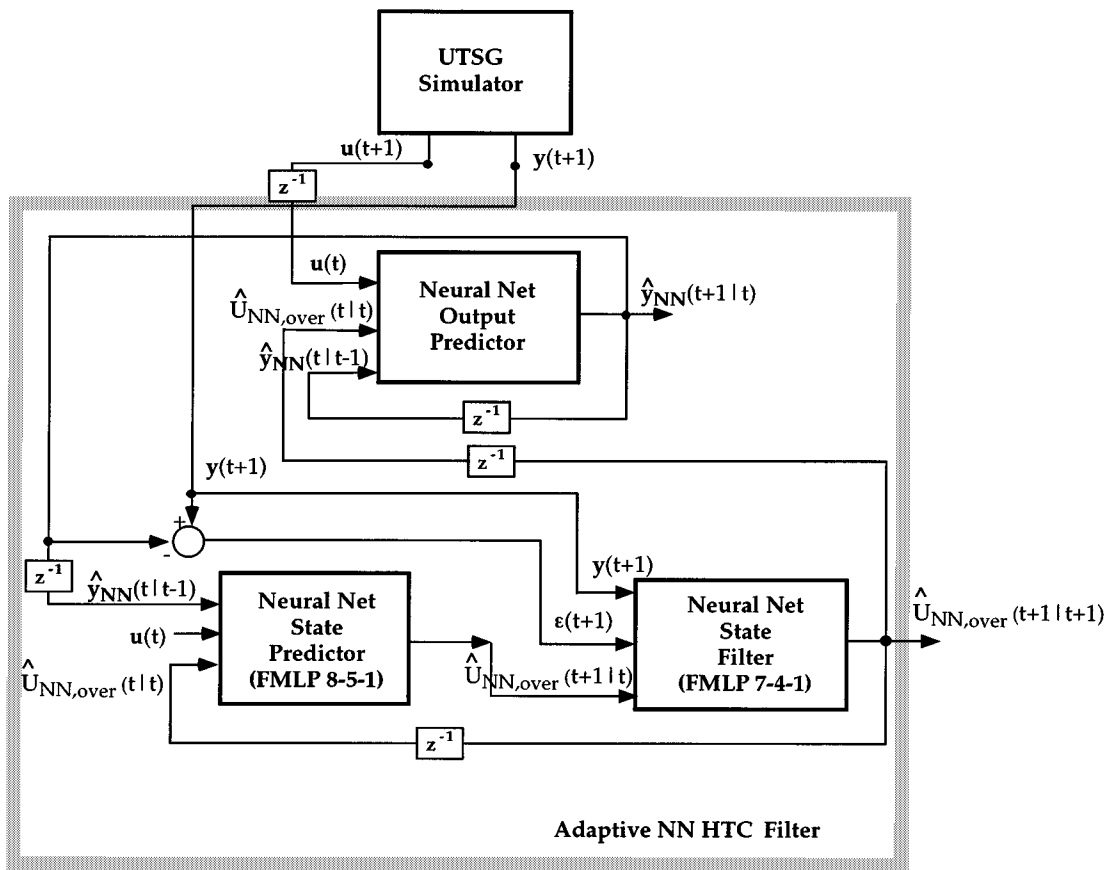


Fig. 6. Block diagram of the UTSG HTC adaptive filter.

TABLE VI
UTSG PROCESS RISER VOID FRACTION ADAPTIVE FILTER VALIDATION
SET ERRORS

Neural State Filter		E_{NMSE}	$\overline{\epsilon_{act}(t)}$	$\overline{\epsilon_{mod}(t)}$	
Adaptive without On-line Learning	Low Noise ($\sigma = 0.05$)	Ramp Input	0.56%	0.38%	0.34%
		Step Input	0.05%	0.18%	0.12%
	High Noise ($\sigma = 0.15$)	Ramp Input	4.5%	-2.6%	-2.48%
		Step Input	0.11%	0.26%	0.24%
Adaptive with On-line Learning	Low Noise ($\sigma = 0.05$)	Ramp Input	0.08%	0.09%	0.54%
		Step Input	0.01%	0.02%	0.45%
	High Noise ($\sigma = 0.15$)	Ramp Input	1.1%	-0.65%	-3.21%
		Step Input	0.01%	0.05%	0.93%

learning, the average state estimation error with respect to the actual states is very small.

Heat Transfer Coefficient Filter: The performance of the HTC adaptive filter is evaluated using the validation data set inputs listed in Table V. Validation of two adaptive filters is presented; without and with on-line learning. Tests 1 to 4 are conducted to evaluate the accuracy of the filtered values of the primary heat transfer coefficient $\hat{U}_{over}(t|t)$ at steady-state UTSG operating power levels. Tests 5 and 6 are conducted to evaluate the performance of the HTC adaptive filter under transient UTSG operating conditions. All tests are performed with low noise corruption while only tests 5 and 6 are performed

with high noise corruption. the average E_{NMSE} for the filtered value of the HTC $\hat{U}_{over}(t|t)$ and the corresponding normalized residual values $\epsilon_{act}(t)$ for the validation data set tests are listed in Table VII. The HTC filtered values for tests 5 and 6, with the data sets corrupted with high levels signal noise, are shown in Figs. 9 and 10, respectively.

Again, considering the difficulties in even computing HTC's off-line using physics-based approaches, the accuracy of the state filters on the validation data set is quite satisfactory. The filter without on-line learning has consistently performed with acceptable performance. The consistency has been observed by considering different type of simulation scenarios. The neural filter with on-line learning further improved the performance of the state filter, by reducing the state filtering errors to very low levels. The improvement in the worst average estimation error is significant; it is reduced from 4% to less than 1%.

6) **EKF's:** In this study, an attempt is made to develop and compare EKF's with the developed neural state filters. The first difficulty in this attempt has been the unavailability of analytical models in state-space form for use in the EKF algorithm. In this study, Choi's simulator [8] is used as the "actual" system for testing the effectiveness of the state filters. The simulator is numerically linearized at many steady-state operating points and the resulting linearized models are scheduled to obtain a piece-wise linear model. The scheduled model is then used in the EKF design. Furthermore, the individual numerically linearized models are used in the EKF error covariance and filter gain calculations. Both the linearized and the scheduled UTSG

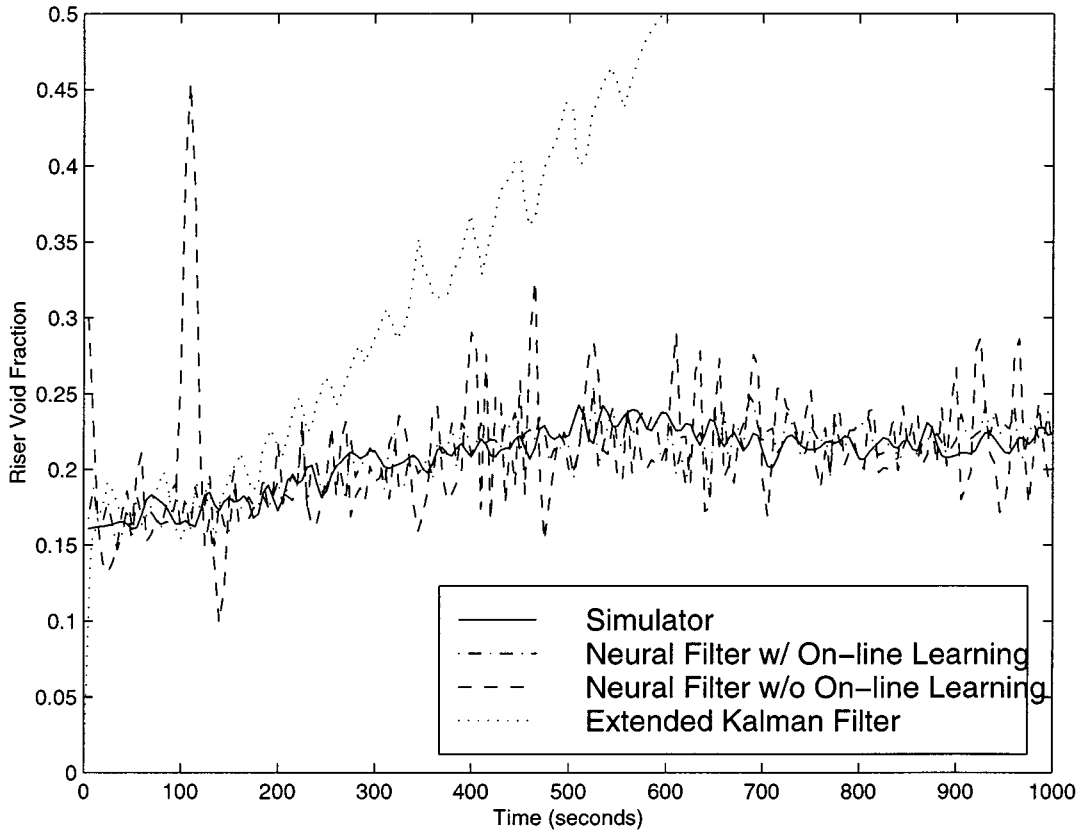


Fig. 7. UTSG process riser void fraction adaptive filter response for a ramp input (high noise environment).

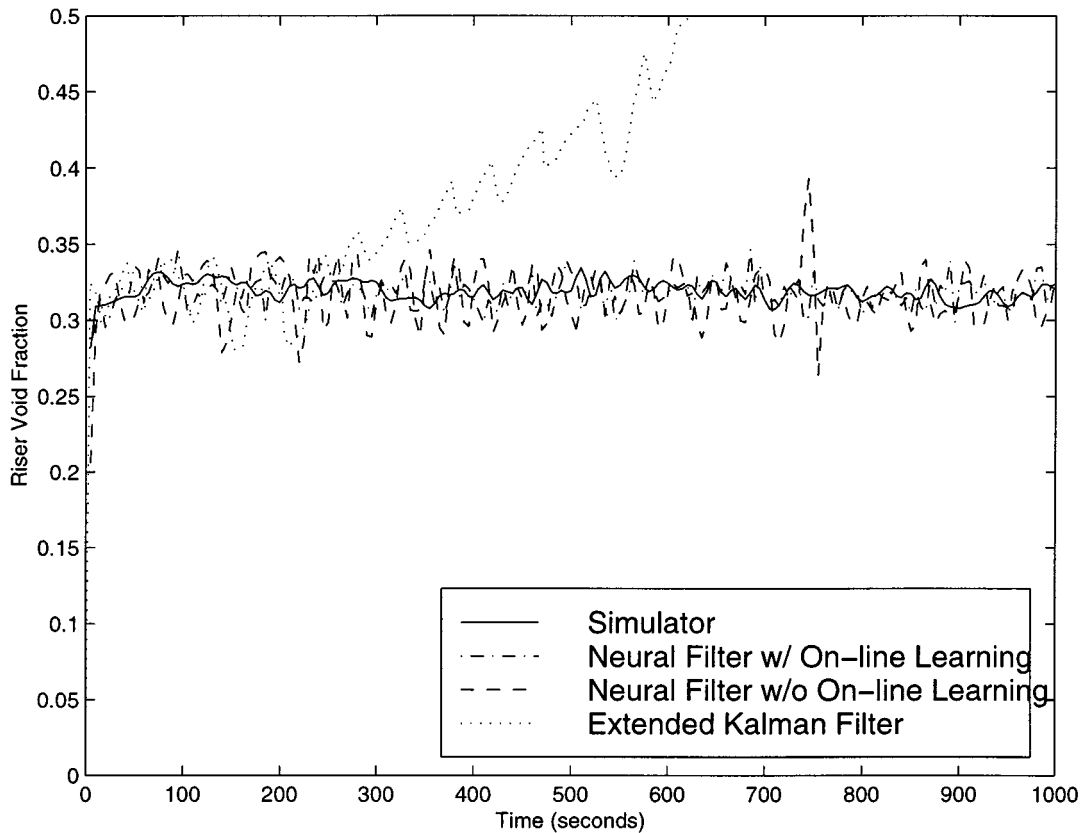


Fig. 8. UTSG process riser void fraction adaptive filter response for a step input (high noise environment).

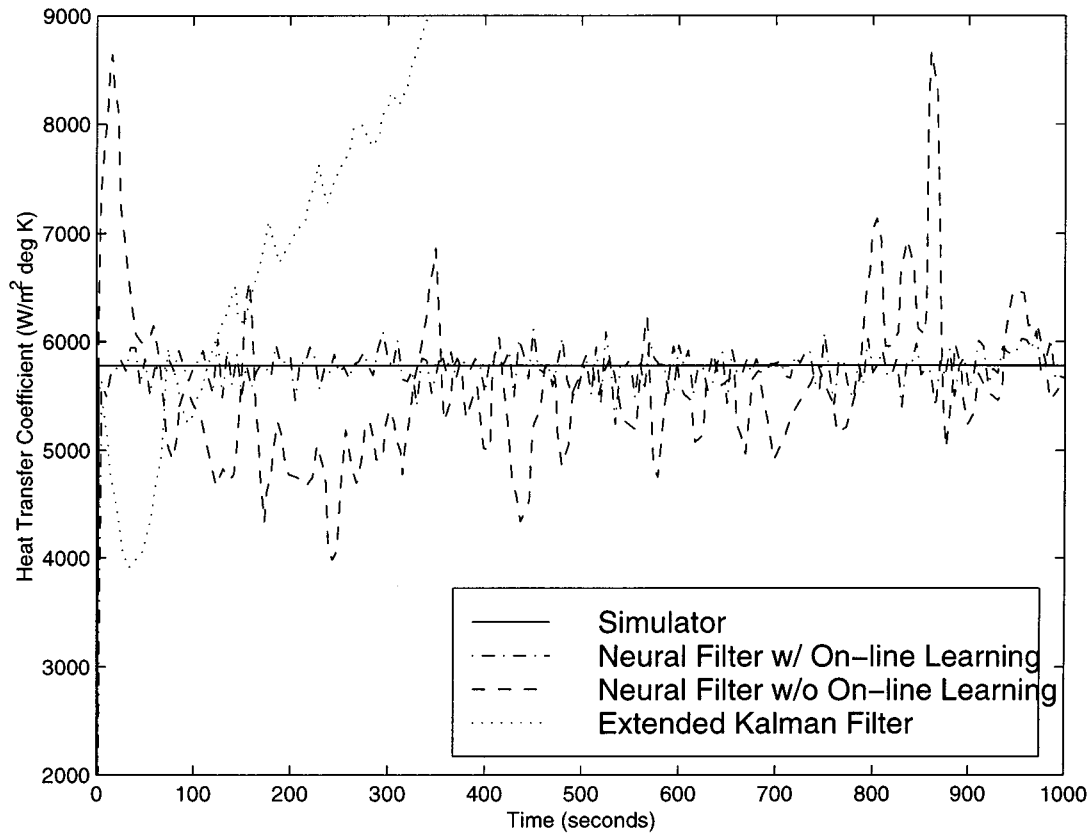


Fig. 9. UTSG process system HTC adaptive filter response using ramp input (high noise environment).

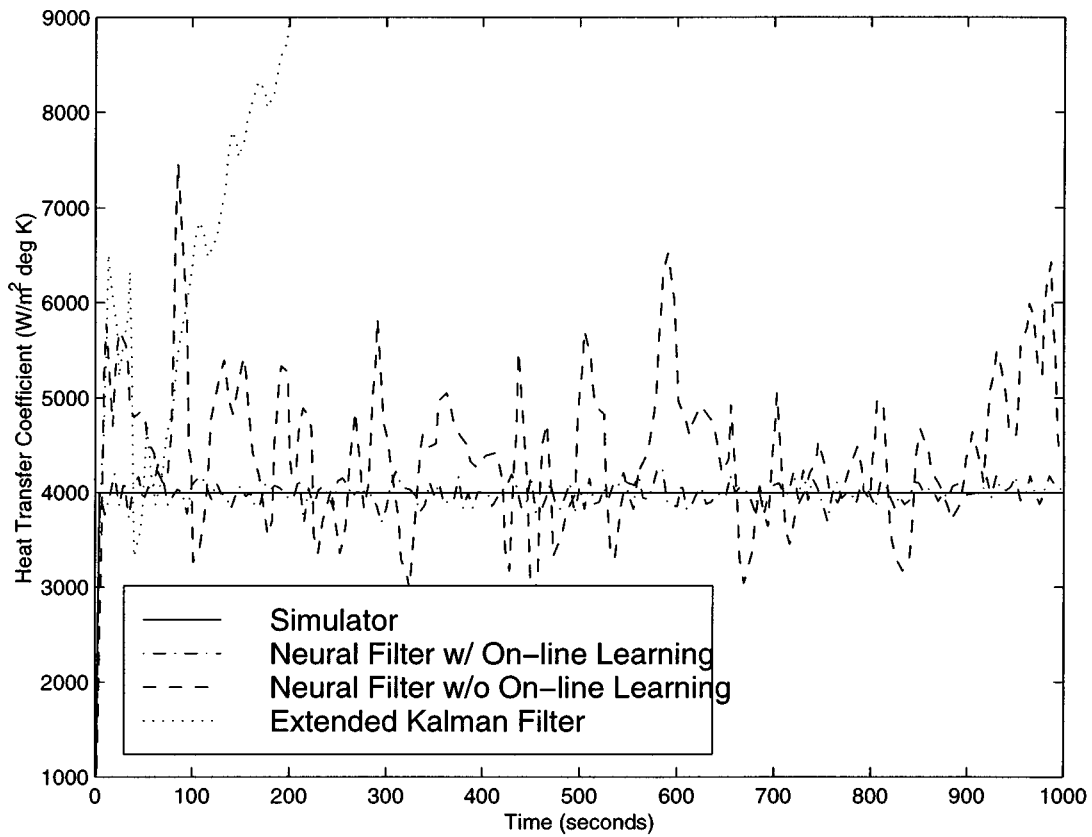


Fig. 10. UTSG process system HTC adaptive filter response using step input (high noise environment).

TABLE VII
UTSG PROCESS HTC ADAPTIVE FILTER VALIDATION SET ERRORS

Neural State Filter		E_{NMSE}	$\overline{\epsilon_{act}(t)}$	
Adaptive without On-line Learning	Low Noise ($\sigma = 0.05$)	Test 1	1.2%	4%
		Test 2	0.8%	2.5%
		Test 3	0.5%	2.7%
		Test 4	0.55%	2.8%
		Test 5	0.46%	1.3%
		Test 6	0.58%	0.7%
		Test 7	0.3%	-0.85%
		Test 8	2.9%	-0.32%
	High Noise ($\sigma = 0.15$)	Test 5	0.6%	1.11%
		Test 6	1.28%	-2.9%
Adaptive with On-line Learning	Low Noise ($\sigma = 0.05$)	Test 1	0.05%	0.9%
		Test 2	0.04%	0.8%
		Test 3	0.05%	0.8%
		Test 4	0.03%	0.7%
		Test 5	0.02%	0.8%
		Test 6	0.03%	0.1%
		Test 7	0.02%	0.09%
		Test 8	0.08%	0.08%
	High Noise ($\sigma = 0.15$)	Test 5	0.06%	0.1%
		Test 6	0.13%	0.6%

models exhibit significant inaccuracies as compared to Choi's simulator. Unfortunately, these represent the only available alternatives for analytical UTSG models for use in the EKF. The errors associated with the numerically linearized and scheduled UTSG models are extensively discussed in Menon and Parlos [27].

After overcoming these serious modeling difficulties, two EKFs are developed for the UTSG; one for the riser void fraction and another for the heat transfer coefficient. Both filters are validated using ramp and step changes in the UTSG operating power level. White Gaussian process and sensor noise is included in the simulations. The riser void fraction filter did not converge to a steady-state value in any of the simulations performed. The state estimates diverged from the actual state values, as shown in Figs. 7 and 8. Similarly no convergence was achieved for the heat transfer coefficient filter also, as shown in Figs. 9 and 10.

Traditionally, EKFs are known to have convergence problems even when analytical nonlinear dynamic models are available. Usually, one would attempt a linearized KF to reduce the impact of convergence problems. However, linearized KFs suffer from performance, especially when the assumed state trajectory is not very accurate. In this case study, linearized KFs did not converge to a steady-state value either. It is believed that the poor performance of the EKFs can be attributed to two reasons: 1) the lack of a nonlinear state-space model that can be analytically differentiated for use in the EKF design and 2) the presence of significant modeling uncertainties in the numerically linearized scheduled model used in developing the EKF, as compared to the "actual" system used in testing the EKF.

VI. SUMMARY AND CONCLUSION

Practical algorithms are presented for nonadaptive and adaptive state filtering in nonlinear dynamic systems using neural networks. The algorithms involve a prediction step and an update step, similar to the EKF framework. In the nonadaptive formulation it is assumed that an accurate system model is available, though not necessarily in analytic form. For the adaptive formulation, an empirical system model is identified prior to filter development. Learning algorithms are presented for the two filters consisting of an off-line and an on-line phase.

Neural state filters address one of the main problems associated with conventional state filtering methods, the limited applicability to real-world problems. Namely, their reliance on perfectly known system models and noise statistics has resulted in limited practical applications. In conventional state filtering methods, even if nonlinear system models are available, some type of model linearization must usually be performed. In the adaptive state filtering algorithms discussed in this paper, the nonlinear dynamics involved in the filter are approximated using neural networks. Therefore, a functional form of the state-space equations is not required. Further, a neural network is used to determine the possibly complex nonlinear mapping between the innovations terms, system outputs, and the state update value. The state filtering algorithms developed in this paper are free from limiting assumptions on the underlying system dynamics and noise statistics, resulting in broad applicability to a large class of problems.

Initially, a simple nonlinear system is chosen to demonstrate the nonadaptive and adaptive neural state filtering algorithms. The two neural filters are developed using an accurate and an inaccurate model of the system, respectively. They are validated using a data set comprised of test signals different than those contained in the estimation data set. Further, the validation data set is corrupted with low and high process noise. An EKF is developed using the model made available to the nonadaptive neural filter for comparison purposes. The EKF state estimation results are comparable to the two neural filters without on-line learning under both low and high noise. All filters have relatively low state estimation error with respect to the model used in their development, whereas more significant error with respect to the actual system states. Further improvements in both neural state estimates with respect to the actual states has resulted from on-line training using additional input and output measurements. This has been at the expense of the state estimation error with respect to the model used in filter design. Despite these results, this case study might not be complex enough to demonstrate a significant benefit for the proposed filtering algorithms.

The second case study is that of a UTSG. This is a complex process system and it is used to develop two adaptive neural state filters and two EKFs. The UTSG system is open-loop unstable and so, for the transients conducted in this study, a controller is used to stabilize it. An adaptive neural filter is developed to estimate the riser void fraction, a state of the UTSG. Additionally, an adaptive neural filter is developed to estimate a constant parameter of the UTSG system, the primary HTC. The EKFs are developed for both these parameters. Furthermore, adaptive estimation results with and without on-line learning are presented.

The two EKF's do not converge to a steady-state value within the simulation time-frame. On the contrary both adaptive neural filters converge quite rapidly. Even without on-line learning the estimation results are acceptable compared to EKF and to the errors obtained in computing these parameters off-line. However, with on-line learning there is significant improvement in the neural state estimates, resulting in estimation errors with respect to the actual states of less than 1%. The results obtained from this case study affirm that the proposed state filtering algorithms can be applied to complex filtering problems with constant or slowly varying parameters. Such critical variables can, among other things, help assess on-line equipment condition for fault diagnosis purposes.

For the two case studies presented it is observed that the accuracy of the filter developed without on-line adaptation is as accurate as the values of the states used in the off-line network training. This couples the obtained state estimation accuracy to the process model accuracy. On-line adaptation has further improved the state estimates, though proper on-line learning algorithms must be used. The success of the filtering algorithms can be partially attributed to the development of separate neural models for the output and state predictors and for the state update. This is only an empirical observation with no theoretical proof or explanation. The results further demonstrate that to the extent possible the state estimation error has been decoupled from the accuracy of the process model used in developing the estimators. If a model is not available for obtaining state values for use in off-line training, a finite sample of state measurements could be collected. Finally, our experience indicates that as the uncertainty and complexity of the state filtering problem increase, the benefit of using the proposed algorithms becomes more apparent. It is only then that complex filters, such as the ones presented in this study, can be justified.

REFERENCES

- [1] A. Alessandri, T. Parisini, and R. Zoppoli, "Neural approximators for nonlinear finite-memory state estimation," *Int. J. Contr.*, vol. 67, no. 2, pp. 275–301, 1997.
- [2] A. M. Annaswamy and S.-H. Yu, " θ -adaptive neural networks: A new approach to parameter estimation," *IEEE Trans. Neural Networks*, vol. 7, July 1996.
- [3] A. Atiya and A. Parlos, "New results on recurrent network training: Unifying the algorithms and accelerating convergence," *IEEE Trans. Neural Networks*, vol. 11, pp. 697–709, 2000.
- [4] A. R. Barron, "Universal approximation bounds for superpositions of a Sigmoidal function," *IEEE Trans. Inform. Theory*, vol. 39, pp. 930–945, 1993.
- [5] —, "Approximation and estimation bounds for artificial neural networks," *J. Machine Learning*, vol. 14, pp. 115–133, 1994.
- [6] S. A. Billings, H. B. Jamaluddin, and S. Chen, "Properties of neural networks with applications to modeling nonlinear dynamical systems," *Int. J. Contr.*, vol. 55, pp. 193–224, 1992.
- [7] M. Boutayeb, H. Rafaralahy, and M. Darouach, "Convergence analysis of the extended Kalman filter used as an observer for nonlinear deterministic discrete-time systems," *IEEE Trans. Automat. Contr.*, vol. 42, pp. 581–586, 1997.
- [8] J. I. Choi, "Nonlinear digital computer control for the steam generator system in a pressurized water reactor plant," Ph.D. dissertation, Massachusetts Inst. Technol., Cambridge, 1987.
- [9] G. Cybenko, "Approximation by superposition of a sigmoidal function," *Math. Contr., Signals, Syst.*, vol. 2, pp. 133–141, June 1990.
- [10] S. Elanayar and Y. C. Shin, "Radial basis function neural network for approximation and estimation of nonlinear stochastic dynamic systems," *IEEE Trans. Neural Networks*, vol. 5, pp. 594–603, July 1994.
- [11] A. Gelb, *Applied Optimal Estimation*. Cambridge, MA: MIT Press, 1974.
- [12] J. J. Gertler, *Fault Detection and Diagnosis in Engineering Systems*. New York: Marcel Dekker, 1998.
- [13] G. C. Goodwin and K. S. Sin, *Adaptive Filtering, Prediction and Control*. Englewood Cliffs, NJ: Prentice-Hall, 1984.
- [14] M. S. Grewal and A. P. Andrews, *Kalman Filtering: Theory and Practice*. Englewood Cliffs, NJ: Prentice-Hall, 1993.
- [15] R. Habtom and L. Litz, "Estimation of unmeasured inputs using recurrent neural networks and the extended Kalman filter," in *Int. Conf. Neural Networks*, vol. 4, 1997, pp. 2067–2071.
- [16] S. Haykin, *Adaptive Filter Theory*, 3rd ed. Upper Saddle River, NJ: Prentice-Hall, 1996.
- [17] S. Haykin, P. Yee, and E. Derbez, "Optimum nonlinear filtering," *IEEE Trans. Signal Processing*, vol. 45, pp. 2774–2786, Nov. 1997.
- [18] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Upper Saddle River, NJ: Prentice-Hall, 1999.
- [19] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359–366, 1989.
- [20] G. Jonsson and O. P. Palsson, "An application of extended Kalman filtering to heat exchanger models," *ASME J. Dyn. Syst., Measurement, Contr.*, vol. 116, pp. 257–264, 1994.
- [21] R. E. Kalman and R. S. Bucy, "New results in linear filtering and prediction theory," *Trans. ASME D. J. Basic Eng.*, vol. 83, pp. 95–107, December 1961.
- [22] J. Lei, H. Guangdong, and J. P. Jiang, "The state estimation of the CSTR system based on a recurrent neural network trained by HGAs," in *Int. Conf. Neural Networks*, vol. 2, 1997, pp. 779–782.
- [23] L. Ljung, *System Identification Theory for the Users*, 2nd ed. Upper Saddle River, NJ: Prentice-Hall, 1999.
- [24] L. Ljung and T. Glad, *Modeling of Dynamic Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1994.
- [25] L. Ljung and T. Sjöberg, "A system identification perspective on neural nets," in *Proc. IEEE Workshop Neural Networks for Signal Processing*, May 1992.
- [26] J. T.-H. Lo, "Synthetic approach to optimal filtering," *IEEE Trans. Neural Networks*, vol. 5, pp. 803–811, Sept. 1994.
- [27] S. K. Menon and A. G. Parlos, "Gain-scheduled nonlinear control of U-tube steam generator water level," *Nucl. Sci. Eng.*, vol. 3, no. 3, pp. 294–308, 1992.
- [28] S. K. Menon, "Adaptive Filtering in Complex Process Systems Using Recurrent Neural Networks," Ph.D. dissertation, Texas A&M University, College Station, TX, 1996.
- [29] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamic system using neural networks," *IEEE Trans. Neural Networks*, vol. 1, pp. 4–27, 1990.
- [30] H. G. Natke and C. Cempel, *Model-Aided Diagnosis of Mechanical Systems*. Berlin: Springer-Verlag, 1997.
- [31] D. Obradovic, "On-line training of recurrent neural networks with continuous topology adaptation," *Trans. Neural Networks*, vol. 7, pp. 222–228, Jan. 1996.
- [32] T. Parisini, A. Alessandri, M. Maggiore, and R. Zoppoli, "On convergence of neural approximate nonlinear state estimators," in *Proc. 1997 Amer. Contr. Conf.*, vol. 3, June 1997, pp. 1819–1822.
- [33] A. G. Parlos, K. T. Chong, and A. Atiya, "Application of the recurrent multilayer perceptron in modeling complex process dynamics," *IEEE Trans. Neural Networks*, vol. 5, pp. 255–266, 1994.
- [34] A. G. Parlos, O. T. Rais, and A. F. Atiya, "Multistep-ahead prediction in complex systems using dynamic recurrent neural networks," *Neural Networks*, 2000, to be published.
- [35] R. Patton, P. M. Frank, and R. N. Clark, *Issues of Fault Diagnosis for Dynamic System*, R. Patton, P. M. Frank, and R. N. Clark, Eds. London, U.K.: Springer-Verlag, 2000.
- [36] K. Reif, S. Gunther, E. Yaz, and R. Unberhauen, "Stochastic stability of the discrete-time extended Kalman filter," *IEEE Trans. Automat. Contr.*, vol. 44, pp. 714–728, 1999.
- [37] I. Rivals and L. Personnaz, "Construction of confidence intervals for neural networks based on least squares estimation," *Neural Networks*, vol. 13, no. 4–5, pp. 463–484, 2000.
- [38] B. Schenker and M. Agarwal, "Predictive control of a bench-scale chemical reactor based on neural-network models," *IEEE Trans. Contr. Syst. Technol.*, vol. 6, pp. 388–400, May 1998.
- [39] F. C. Schweppe, *Uncertain Dynamic Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1973.
- [40] T. Soderstrom and S. Stoica, *System Identification*. Englewood Cliffs, NJ: Prentice-Hall, 1988.

- [41] W. H. Strohmayer, "Dynamic Modeling of Vertical U-Tube Steam Generators for Operational Safety Systems," Ph.D. dissertation, MIT, Cambridge, MA, 1982.
- [42] S. C. Stubberud, R. N. Lobbia, and M. Owen, "An adaptive extended Kalman filter using artificial neural networks," in *Proc. 34th IEEE Conf. Decision Contr.*, vol. 2, Dec. 1995, pp. 1852–1856.
- [43] S. C. Stubberud and M. Owen, "Targeted on-line modeling for an extended Kalman filter using artificial neural networks," in *Proc. 1998 Amer. Contr. Conf.*, vol. 3, June 1998, pp. 1852–1856.
- [44] J. A. K. Suykens, B. L. R. De Moor, and J. Vandewalle, "Nonlinear system identification using neural state-space models, applicable to robust control design," *Int. J. Cont.*, vol. 62, no. 1, pp. 129–152.
- [45] R. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Comput.*, vol. 1, pp. 270–280, 1989.
- [46] R. Zhu, T. Chai, and C. Shao, "Robust nonlinear adaptive observer design using dynamic recurrent neural networks," in *Proc. 1997 Amer. Contr. Conf.*, vol. 2, June 1997, pp. 1096–1100.

Alexander G. Parlos (S'81–M'87–SM'92) received the B.S. degree in nuclear engineering from Texas A&M University, College Station, in 1983. He received the S.M. degree in mechanical engineering, the S.M. degree in nuclear engineering, and the Sc.D. degree in automatic control and systems engineering, all from Massachusetts Institute of Technology, Cambridge, in 1985, 1985, and 1986, respectively.

He has been on the faculty at Texas A&M University since 1987, where he is currently an Associate Professor of Mechanical Engineering, with joint appointments in the Department of Nuclear Engineering and Department of Electrical Engineering. His applied research interests include the development of methods and algorithms for life-cycle health and performance management of various dynamic systems, with special emphasis to system condition assessment (or diagnosis), end-of-life prediction (or prognosis) and reconfigurable control. He has been involved with the particular application of these concepts to electro-mechanical systems and more recently to computer networks. His theoretical-research interests involve the development of learning algorithms for recurrent neural networks and their use for nonlinear estimation and control. He has been involved with research and teaching in neural networks, multivariable control, and system identification and he has conducted extensive funded research in these areas. His research has resulted in one United States patent, three pending patents, and 18 invention disclosures. He has cofounded a high-tech startup company commercializing technology developed at Texas A&M. He has more than 135 publications in journals and conferences.

Dr. Parlos has served as a technical reviewer to numerous professional journal and government organizations and he has participated in technical, organizing and program committees of various conferences. He has served as an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS since 1994 and of the *Journal of Control, Automation and Systems* since 1999. He is a Senior Member of AIAA, a member of ASME, ANS, INNS, and a registered Professional engineer in the state of Texas.

Sunil K. Menon (S'90–M'97) received the B.E. degree with Honors in electrical and electronics engineering from the Regional Engineering College, Tiruchirapalli, India, in 1988 and the M.S. and Ph.D. degrees in nuclear engineering from Texas A&M University, College Station, in 1990 and 1996, respectively.

In 1996, he joined Honeywell Laboratories, Minneapolis, MN, and is currently working as a Principal Research Scientist in the Information and Decision Technology Laboratory. His primary interests include neural networks, system fault detection, diagnosis and prognosis, nonlinear system identification, and multivariate control.

Dr. Menon is a member of the ASME and has served as a reviewer of papers for various journals and conferences.



Amir F. Atiya (S'86–M'90–SM'97) was born in Cairo, Egypt, in 1960. He received the B.S. degree in 1982 from Cairo University and the M.S. and Ph.D. degrees in 1986 and 1991 from California Institute of Technology (Caltech), Pasadena, all in electrical engineering.

From 1985 to 1990, he was a Teaching and Research Assistant at Caltech. From September 1990 to July 1991, he held a Research Associate position at Texas A&M University. From July 1991 to February 1993, he was a Senior Research Scientist at QANTXX, Houston, TX, a financial modeling firm. In March 1993, he joined the Computer Engineering Department at Cairo University as an Assistant Professor. He spent the summer of 1993 with Tradelink Inc., Chicago, IL, and the summers of 1994, 1995, and 1996 with Caltech. Since March 1997, he has been a Visiting Associate in the Department of Electrical Engineering at Caltech. In addition, from January 1998 to January 2000, he was a Research Scientist at Simplex Risk Management Inc., Hong Kong. His research interests are in the areas of neural networks, learning theory, pattern recognition, Monte Carlo methods, data compression, and optimization theory. His most recent interests are the application of learning theory and computational methods to finance. He has also been active in developing quantitative and machine learning-based financial market trading systems.

Dr. Atiya received the highly regarded *Egyptian State Prize for Best Research in Science and Engineering*, in 1994. He also received the *Young Investigator Award from the International Neural Network Society*, in 1996. He has been an Associate Editor for the IEEE TRANSACTIONS ON NEURAL NETWORKS since 1998. He was a Co-Guest Editor of a special issue of IEEE TRANSACTIONS ON NEURAL NETWORKS on Neural Networks in Financial Engineering. He served on the organizing and program committees of several conferences, most recent of which is Computational Finance CF2000, New York.