

An Adaptive State Filtering Algorithm for Systems With Partially Known Dynamics

Alexander G. Parlos

Department of Mechanical Engineering,
Texas A&M University,
College Station, TX 77843
e-mail: a-parlos@tamu.edu

Sunil K. Menon

Honeywell Technology Center,
3660 Technology Drive,
MN65-2500, Minneapolis, MN 55418
e-mail: sunil@htc.honeywell.com

Amir F. Atiya

Department of Electrical Engineering,
California Institute of Technology,
MS 136-93, Pasadena, CA 91125
e-mail: amir@work.caltech.edu

On-line filtering of stochastic variables that are difficult or expensive to directly measure has been widely studied. In this paper a practical algorithm is presented for adaptive state filtering when the underlying nonlinear state equations are partially known. The unknown dynamics are constructively approximated using neural networks. The proposed algorithm is based on the two-step prediction-update approach of the Kalman Filter. The algorithm accounts for the unmodeled nonlinear dynamics and makes no assumptions regarding the system noise statistics. The proposed filter is implemented using static and dynamic feed-forward neural networks. Both off-line and on-line learning algorithms are presented for training the filter networks. Two case studies are considered and comparisons with Extended Kalman Filters (EKFs) performed. For one of the case studies, the EKF converges but it results in higher state estimation errors than the equivalent neural filter with on-line learning. For another, more complex case study, the developed EKF does not converge. For both case studies, the off-line trained neural state filters converge quite rapidly and exhibit acceptable performance. On-line training further enhances filter performance, decoupling the eventual filter accuracy from the accuracy of the assumed system model. [DOI: 10.1115/1.1485747]

Keywords: Adaptive State Filtering, Dynamic Neural Networks, Nonlinear State Filtering, Multi-Step Prediction, Extended Kalman Filtering

1 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. Since the development of the well-known KF approach for linear stochastic state estimation [1], this method has been widely studied in the literature and applied to many problems. The KF has been extended to nonlinear systems. One such extension, called the EKF [2], has experienced a lot of interest from researchers but it has found few practical industrial applications due to difficulties in obtaining convergent state estimates [3].

Neural networks have been extensively investigated in the context of adaptive control and system identification [4], 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 [5]. Elanayar and Shin used radial basis function neural networks to investigate state estimation problems [6]. Also recently, the problem of parameter estimation has received the attention of the neural networks community with some new results reported [7]. Nonlinear, finite-memory state estimators using neural networks have been proposed by Parisini et al. [8] and Alessandri et al. [9], in related publications. Haykin et al. has presented an overview of state filtering with neural networks, with emphasis on radial basis function networks which are linear-in-parameters [10]. Zhu et al. proposed the design of an adaptive observer using dynamic neural networks [11], and Habtom inves-

tigated the use of the EKF with an identified neural network model [12]. Dong et al. use neural networks for adaptive filters in an on-line fault detection scheme [13]. Lei et al. used recurrent neural networks to estimate the state of a chemical process [14], whereas Schenker and Agarwal used neural networks for predictive control of a chemical reactor that involved state estimation [15]. Stubberud et al. used neural networks to implement an EKF, comparing various implementation options [16,17]. Finally, Durovic and Kovacevic have used recurrent neural networks for adaptive state filtering that is based on deterministic observer theory [18].

In this paper, a practical algorithm is presented for effective adaptive state filtering in systems with partially known nonlinear dynamics. In previous work, the authors developed an adaptive state filtering algorithm using recurrent neural networks for systems with completely unknown nonlinear dynamics; complete identification of an empirical model was part of the adaptive filter [19,20]. The current paper makes the following contributions:

- A practical algorithm is presented for adaptive state filtering in nonlinear systems with partially known dynamics. Our formulation is based on the recursive approach of the EKF. However, unlike the EKF, the case of partially known state dynamics is considered. Separate neural networks models are developed for each "mapping" involved in the filtering algorithm.
- The effectiveness of the presented algorithm is demonstrated by developing state filters for a motor-pump system and a complex process, namely a UTSG. The performance of the developed neural filters is extensively tested and compared to EKF, demonstrating the resulting improvements in performance and broad applicability. For the motor-pump system the online trained adaptive filter reveals much lower average state estimation errors than the EKF. For the case of the UTSG, the adaptive state filter estimation error is quite acceptable while the EKF fails to converge.

Contributed by the Dynamic Systems and Control Division for publication in the JOURNAL OF DYNAMIC SYSTEMS, MEASUREMENT, AND CONTROL. Manuscript received by the Dynamic Systems and Control Division December 2000. Associate Editor: R. Langari.

2 Nonlinear State Filtering

2.1 Problem Statement. Consider the following system representation in discrete-time, nonlinear, stochastic state-space form, also known as the *noise representation*

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

where $k=1,2,\dots$ is the discrete time instant. Further, it is assumed that $\mathbf{w}(k)$ and $\mathbf{v}(k)$ are independent stochastic processes. At this stage no assumptions are made regarding implicit or explicit knowledge of $\mathbf{f}(\cdot)$ and $\mathbf{h}(\cdot)$, other than assuming 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 $\mathbf{w}(k)$ and $\mathbf{v}(k)$ are considered zero-mean, white Gaussian processes. Equations (1) are the starting point for our development.

The objective of the state filtering problem is to obtain an estimate, $\hat{\mathbf{x}}(k)$, for the state $\mathbf{x}(k)$ 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 $\hat{\mathbf{x}}(k|k)$ to simply mean the state estimate at time k , following the update resulting from the measurements $\mathbf{u}(k)$ and $\mathbf{y}(k)$, at time k .

2.2 Conventional Method of Solution. The EKF solution to nonlinear state filtering assumes the availability of an exact system model depicted by equations (1). A summary of the algorithm consists of the following two steps [2,21]:

Step 1-Prior to Observing the $(k+1)^{\text{th}}$ Sample-Prediction Step: The assumed model $\mathbf{f}(\cdot)$ and $\mathbf{h}(\cdot)$, and the state estimate at a given time step, $\hat{\mathbf{x}}(k|k)$, are used to compute the predicted value of the state and output, $\hat{\mathbf{x}}(k+1|k)$ and $\hat{\mathbf{y}}(k+1|k)$, 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 used in this step.

Step 2-Following Observation of $(k+1)^{\text{th}}$ Sample-Update Step: The state prediction, $\hat{\mathbf{x}}(k+1|k)$, is updated using a linear combination of the prediction errors (or innovations), $\mathbf{y}(k+1)$

$-\hat{\mathbf{y}}(k+1|k)$, resulting in the state estimate $\hat{\mathbf{x}}(k+1|k+1)$. The coefficients used to weigh the innovations terms form the elements of the EKF gain matrix. The EKF gain matrix is updated using the Jacobian of $\mathbf{h}(\cdot)$ evaluated at the state prediction, 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.

2.3 Proposed 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 replaced by the combination of an assumed deterministic states-space system model, $\mathbf{f}_{\text{mod}}(\cdot)$ and $\mathbf{h}_{\text{mod}}(\cdot)$, in the form of an open-loop predictor or simulation model [22], and two neural networks used in approximating the significant dynamics absent from the assumed model; these two neural networks are referred to as the “error model” (EM),
- the functional form of the filter state update equation, or the filter gain, is allowed to be nonlinear, say $\mathcal{K}(\cdot)$, and it is also constructed from input/output measurements, and,
- the process and sensor noise statistics are assumed unknown and they are not explicitly used in the filter computations.

3 Adaptive Neural State Filter

In this section we formulate the adaptive state filter. The assumed system model and the neural network EM are explicitly utilized in the filter computations.

3.1 Filter Equations. To obtain a state estimate, $\hat{\mathbf{x}}(k+1|k+1)$, using a nonlinear system model of the form given by equations (1) and an identified EM, the following prediction-update algorithm is proposed:

Step 1 - Prior to Observing the $(k+1)^{\text{th}}$ Sample - Prediction Step:

The state prediction is computed using the two steps

$$\begin{cases} \hat{\mathbf{x}}(k+1|k) = \mathbf{f}_{\text{mod}}(\hat{\mathbf{x}}_{\text{NN}}(k|k), \mathbf{u}(k)), \\ \hat{\mathbf{x}}_{\text{NN}}(k+1|k) = \mathbf{f}_{\text{NN}}(\hat{\mathbf{x}}(k+1|k), \mathcal{Y}(k), \mathcal{U}(k), \mathcal{E}_x(k|k-1)), \end{cases} \quad (2)$$

where the vectors involved in Eq. (2) are defined as

$$\begin{cases} \mathcal{Y}(k) \equiv [\mathbf{y}(k), \mathbf{y}(k-1), \dots, \mathbf{y}(k-n_y+1)]^T, \\ \mathcal{U}(k) \equiv [\mathbf{u}(k), \mathbf{u}(k-1), \dots, \mathbf{u}(k-n_u+1)]^T, \\ \mathcal{E}_x(k|k-1) \equiv [\boldsymbol{\epsilon}_x(k|k-1), \boldsymbol{\epsilon}_x(k-1|k-2), \dots, \boldsymbol{\epsilon}_x(k-n_x^x+1|k-n_x^x)]^T, \end{cases} \quad (3)$$

and where,

$$\boldsymbol{\epsilon}_x(k|k-1) \equiv \hat{\mathbf{x}}(k|k-1) - \hat{\mathbf{x}}_{\text{NN}}(k|k-1). \quad (4)$$

The number of delays, n_y , n_u and n_e^x are determined iteratively during the network training stage of filter construction.

The output prediction is computed using two similar steps

$$\begin{cases} \hat{\mathbf{y}}(k+1|k) = \mathbf{h}_{\text{mod}}(\hat{\mathbf{x}}(k+1|k), \mathbf{u}(k)), \\ \hat{\mathbf{y}}_{\text{NN}}(k+1|k) = \mathbf{h}_{\text{NN}}(\hat{\mathbf{y}}(k+1|k), \mathcal{Y}(k), \mathcal{U}(k), \mathcal{E}_y(k|k-1)), \end{cases} \quad (5)$$

where

$$\begin{aligned} \mathcal{E}_y(k|k-1) \equiv & [\boldsymbol{\epsilon}_y(k|k-1), \boldsymbol{\epsilon}_y(k-1|k-2), \dots, \boldsymbol{\epsilon}_y(k-n_e^y \\ & + 1|k-n_e^y)]^T, \end{aligned} \quad (6)$$

and where,

$$\boldsymbol{\epsilon}_y(k|k-1) \equiv \hat{\mathbf{y}}(k|k-1) - \hat{\mathbf{y}}_{\text{NN}}(k|k-1). \quad (7)$$

As in the previous predictor, number of delays n_e^y is determined iteratively during the network training stage of filter construction.

Step 2 - Following Observation of $(k+1)^{\text{th}}$ Sample - Update Step:

The state is updated using the state and output prediction of the EM, as follows:

$$\hat{\mathbf{x}}_{\text{NN}}(k+1|k+1) = \mathcal{K}_{\text{NN}}(\hat{\mathbf{x}}_{\text{NN}}(k+1|k), \mathcal{Y}(k+1), \mathcal{E}(k+1)), \quad (8)$$

where $\mathcal{E}(k+1)$ is defined as

$$\mathcal{E}(k+1) \equiv [\boldsymbol{\epsilon}(k+1), \boldsymbol{\epsilon}(k), \dots, \boldsymbol{\epsilon}(k-n_e+1)]^T, \quad (9)$$

and where,

$$\boldsymbol{\epsilon}(k+1) \equiv \mathbf{y}(k+1) - \hat{\mathbf{y}}_{\text{NN}}(k+1|k). \quad (10)$$

The nonlinear filter gain, $\mathcal{K}_{\text{NN}}(\cdot)$, is the output of a neural network constructed at the filter design stage.

Figure 1 shows a block diagram of the proposed adaptive state filter. In this formulation, there are three neural networks to be constructed, $\mathbf{f}_{NN}(\cdot)$, $\mathbf{h}_{NN}(\cdot)$, and $\mathcal{K}_{NN}(\cdot)$. The networks considered for use are static and dynamic feedforward neural networks [4,23], with well-documented approximation properties [24]. Static networks have the structure of FIR-type filters, and training must be performed using all network inputs sampled from the training set, the so-called TF [4]. Similarly, dynamic networks have the structure of IIR-type filters, and training must be performed using some network inputs from the training set, whereas others (delayed outputs) using tapped delays (past predictions), the so-called GF [4,25]. In training a dynamic network, initially the network predictions are inaccurate and this prevents effective training. In this study, dynamic networks are initially trained using TF, followed by GF [25].

3.2 The Off-Line Training Phase. In the off-line training phase of the filter, in addition to the output measurements used in the training set, $\mathbf{y}_{\text{target}}$, it is assumed that some reliable state values, $\mathbf{x}_{\text{target}}$, are also available for use in training. The latter can be obtained from experiments or from the assumed system model, $\mathbf{f}_{\text{mod}}(\cdot)$ and $\mathbf{h}_{\text{mod}}(\cdot)$. The former 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 continuously. The effectiveness of the latter assumption about state values depends on the accuracy of the available system model. In some instances, even off-line signal processing methods can be used for extracting state values from input and output measurements.

The off-line training is divided into two phases. In the first phase all filter networks are trained as if they were static; that is, they are trained separately using TF. In the second phase the dynamic networks of the filter are coupled and further trained using GF. In the first phase of off-line learning, \mathbf{h}_{NN} is developed first, followed by \mathbf{f}_{NN} , and then the \mathcal{K}_{NN} network. The following objective functions are minimized in this phase of the training process

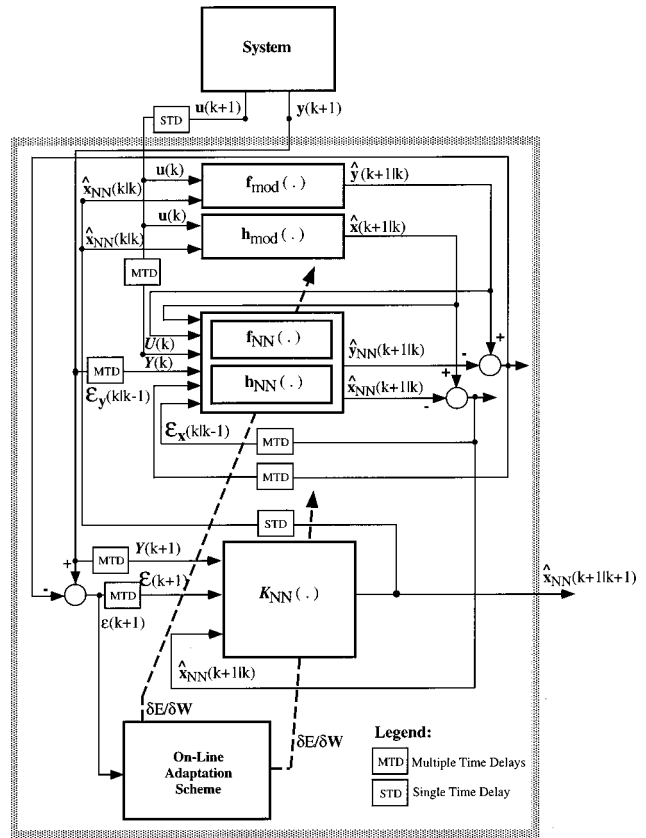


Fig. 1 Block diagram of the neural network state filter

$$\begin{cases} E_1 \equiv \sum_{k=0}^{NP-1} E_1(k+1) \equiv \sum_{k=0}^{NP-1} \sum_{\ell=1}^K [\hat{y}_{NN,\ell}(k+1|k) - y_{\text{target},\ell}(k+1)]^2, \\ E_2 \equiv \sum_{k=0}^{NP-1} E_2(k+1) \equiv \sum_{k=0}^{NP-1} \sum_{\ell=1}^K [\hat{x}_{NN,\ell}(k+1|k) - x_{\text{target},\ell}(k+1)]^2, \\ E_3 \equiv \sum_{k=0}^{NP-1} E_3(k+1) \equiv \sum_{k=0}^{NP-1} \sum_{\ell=1}^K [\hat{x}_{NN,\ell}(k+1|k+1) - x_{\text{target},\ell}(k+1)]^2, \end{cases} \quad (11)$$

The error gradients for feedforward networks trained with TF can be obtained by direct application of the chain-rule. The detailed computation of these gradients can be found in many neural networks references, such as [4,26,27].

Following completion of the first phase of the off-line learning, GF is used to further train the dynamic networks involved in the state filter, that is the \mathbf{f}_{NN} and \mathbf{h}_{NN} networks. This form of learning necessitates minimization of a MSP error objective function over a possibly moving window, W_s , as follows [25]:

$$\sum_{k=k_0+1}^{k_0+W_s} E(k_0, k) \equiv \sum_{k=k_0+1}^{k_0+W_s} \sum_{\ell=1}^K (\hat{y}_{\ell}(k|k_0) - y_j(k))^2, \quad (12)$$

where the error $E(k_0, k)$ depends both on the window location and the prediction point within the window. For off-line learning we select $k_0=1$ and $W_s=NP-k_0$. In this phase, the two predictors and the state update network of the filter are further trained in

tandem, by using the response of one network to improve the predictive response of the two others, until all networks produce acceptably accurate MSP responses, as determined by a cross-validation set. The only measurements used as network inputs in this phase of the off-line training are the delayed system inputs, $\mathbf{u}(k)$. 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 this phase are omitted, but they can be found in a recent publication [25]. More recent recurrent learning algorithms can also be considered in this phase of the training that can accelerate the learning process considerably [28].

3.3 The On-Line Training Phase. Neural networks are “semi-parametric” empirical models and therefore they can be “retrained” recursively, on-line training, as new observations become available. This training phase is complicated by the fact that

the only measurements available are those of the system inputs and outputs, $\mathbf{u}(k)$ and $\mathbf{y}(k)$, respectively. No state information is available on-line.

As in the first phase of the off-line training, during on-line training the networks are decoupled and treated separately. For all the filter networks, only a SSP gradient propagation training is considered because on-line learning with MSP minimization becomes exceedingly complex and impractical. During on-line training, the error function to be minimized is of the form

$$E(k+1) \equiv \sum_{\ell=1}^K [\hat{\mathbf{y}}_{NN,\ell}(k+1|k) - \mathbf{y}_{\text{target},\ell}(k+1)]^2. \quad (13)$$

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

$$\frac{\partial E(k+1)}{\partial \mathbf{w}_{\mathbf{h}_{NN}}} = 2[\hat{\mathbf{y}}_{NN}(k+1|k) - \mathbf{y}_{\text{target}}(k+1)]^T \frac{\partial \mathbf{h}_{NN}}{\partial \mathbf{w}_{\mathbf{h}_{NN}}}. \quad (14)$$

For the state predictor network, the equivalent on-line training gradient can be expressed as

$$\begin{aligned} \frac{\partial E(k+1)}{\partial \mathbf{w}_{\mathbf{f}_{NN}}} &= 2[\hat{\mathbf{y}}_{NN}(k+1|k) - \mathbf{y}_{\text{target}}(k+1)]^T \left(\frac{\partial \mathbf{h}_{NN}}{\partial \hat{\mathbf{y}}(k+1|k)} \right) \\ &\quad \times \left(\frac{\partial \mathbf{h}_{\text{mod}}}{\partial \hat{\mathbf{x}}(k+1|k)} \right) \left(\frac{\partial \mathbf{f}_{\text{mod}}}{\partial \hat{\mathbf{x}}_{NN}(k|k)} \right) \left(\frac{\partial \mathcal{K}_{NN}}{\partial \hat{\mathbf{x}}_{NN}(k/k-1)} \right) \\ &\quad \times \left(\frac{\partial \mathbf{f}_{NN}}{\partial \mathbf{w}_{\mathbf{f}_{NN}}} \right). \end{aligned} \quad (15)$$

Finally, for the state update network the error gradient can be expressed as

$$\begin{aligned} \frac{\partial E(k+1)}{\partial \mathbf{w}_{\mathcal{K}_{NN}}} &= 2[\hat{\mathbf{y}}_{NN}(k+1|k) - \mathbf{y}_{\text{target}}(k+1)]^T \left(\frac{\partial \mathbf{h}_{NN}}{\partial \hat{\mathbf{y}}(k+1|k)} \right) \\ &\quad \times \left(\frac{\partial \mathbf{h}_{\text{mod}}}{\partial \hat{\mathbf{x}}(k+1|k)} \right) \left(\frac{\partial \mathbf{f}_{\text{mod}}}{\partial \hat{\mathbf{x}}_{NN}(k|k)} \right) \left(\frac{\partial \mathcal{K}_{NN}}{\partial \mathbf{w}_{\mathcal{K}_{NN}}} \right). \end{aligned} \quad (16)$$

In Eqs. (15) and (16), the contributions of the terms $\boldsymbol{\epsilon}_x(\cdot)$ and $\boldsymbol{\epsilon}_y(\cdot)$ to the on-line gradients are ignored.

Some of the gradients contained in Eqs. (14) through (16) can be computed using a sensitivity-type network once the specific neural network architecture is selected. The gradients of $\mathcal{K}_{NN}(\cdot)$ with respect to $\mathbf{w}_{\mathcal{K}_{NN}}$ also depend on the specific architecture of the network used, and for a feedforward network they can be obtained using a backpropagation-type procedure. 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 an analytical form of the assumed model is available.

3.4 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 filter networks. Applying only off-line training will usually not be sufficient. This can be seen by observing Eq. (8), and focusing on the input $\hat{\mathbf{x}}_{NN}(k+1|k)$. For the off-line case, $\hat{\mathbf{x}}_{NN}(k+1|k)$ is given as $\mathbf{f}_{\text{mod}}(\mathbf{x}_{\text{target}}(k), \mathbf{u}(k))$ and the neural network, $\mathcal{K}_{NN}(\cdot)$, essentially attempts to infer the value of $\mathbf{x}_{\text{target}}(k+1)$ given the values of $\mathbf{x}_{\text{target}}(k)$ and $\mathbf{y}(k+1)$. On the other hand, for the on-line case the network $\mathcal{K}_{NN}(\cdot)$ attempts to infer $\mathbf{x}_{\text{target}}(k+1)$ given $\mathbf{y}(k+1)$, and the previous estimate $\hat{\mathbf{x}}_{NN}(k|k)$, which is not exactly the same as $\mathbf{x}_{\text{target}}(k)$. Therefore, on-line training is more realistic and indeed more accurate than off-line training, allowing the filter to operate in closed-loop form.

Filter performance is assessed using the following errors:

$$\boldsymbol{\epsilon}_{\text{act}}(k) = \frac{\mathbf{x}(k) - \hat{\mathbf{x}}(k|k)}{\mathbf{x}(k)}, \quad \boldsymbol{\epsilon}_{\text{mod}}(k) = \frac{\mathbf{x}_m(k) - \hat{\mathbf{x}}(k|k)}{\mathbf{x}_m(k)}. \quad (17)$$

The former is a measure of the state estimate accuracy with respect to the ‘‘simulated’’ state value, whereas the latter is a measure of the state estimate accuracy with respect to the state value computed from the assumed model used in off-line training.

4 A Motor-Pump System Case Study

The objective of this case study is to develop an adaptive state filter for simultaneous estimation of a DC motor armature resistance and flux linkage.

4.1 Motor-Pump System Description. The motor-pump system comprises a DC motor, a centrifugal pump, and the associated piping system. The equations governing the operation of the system have been derived by closely following the approaches of [29,30]. The armature circuit is expressed as

$$L_a \frac{dI_a(t)}{dt} + R_a(t)I_a(t) + K_v \Psi \omega(t) = V_a. \quad (18)$$

The armature resistance is assumed to vary with the armature current, or equivalently with the external load demand, according to

$$R_a(t) = \alpha \frac{dI_a^2(t)}{dt} + R_a^o. \quad (19)$$

The angular velocity of the combined DC motor and the centrifugal pump is described by

$$\theta_{mp} \frac{d\omega(t)}{dt} = \Psi I_a(t) - c_{mp0} - c_{mp1}\omega(t) - h_2 \dot{M}(t)\omega(t). \quad (20)$$

The dynamic equation for the mass flow rate is described by

$$\frac{d\dot{M}(t)}{dt} = h_{nn}\dot{M}^2(t) - h_{rr}(t)\dot{M}^2(t). \quad (21)$$

The external load on the overall system is varied through variations in $h_{rr}(t)$. The external load cannot be directly measured and therefore it must be estimated. It is assumed that three of the motor-pump system states, $I_a(t)$, $\omega(t)$, and $\dot{M}(t)$, are measured. On the other hand, the armature resistance, $R_a(t)$, cannot be directly measured and it must be estimated. The set of Eqs. (18) through (21) comprises the motor-pump simulator.

Additionally, an assumed motor-pump model is implemented by replacing Eq. (21) with

$$\dot{M}(t) = K_{mM}\omega(t). \quad (22)$$

Equations (18) through (20) are also used in the assumed motor-pump model. However, the physical parameters used in the assumed model are different than those used in the motor-pump simulator. The $\hat{T}_L(t)$ is estimated from the measured signals, $\omega(t)$ and $\dot{M}(t)$, using the following semi-empirical relation

$$\hat{T}_L(t) = k\dot{M}(t)\omega(t), \quad (23)$$

where $k=0.36$ is an empirically determined constant. The $\hat{T}_L(t)$ is one of the inputs used in the neural state filter. The equations describing the predictor form of the assumed motor-pump model are

$$\begin{cases} \hat{I}_a(k+1|k) = f_{\text{mod}}^{mp1}(\hat{I}_a(k|k), \hat{\omega}(k|k), \hat{R}_a(k|k)), \\ \hat{R}_a(k+1|k) = f_{\text{mod}}^{mp2}(\hat{I}_a(k|k)), \\ \hat{\omega}(k+1|k) = f_{\text{mod}}^{mp3}(\hat{I}_a(k|k), \hat{\omega}(k|k), \hat{T}_L(k)), \\ \hat{M}(k+1|k) = f_{\text{mod}}^{mp4}(\hat{\omega}(k|k)), \end{cases} \quad (24)$$

where the functions $\mathbf{f}_{\text{mod}} = [f_{\text{mod}}^{mp1}(\cdot), f_{\text{mod}}^{mp2}(\cdot), f_{\text{mod}}^{mp3}(\cdot), f_{\text{mod}}^{mp4}(\cdot)]^T$ are the discrete-time equivalents of the assumed motor-pump model. The discrete-time motor-pump model is obtained by applying the standard discretization transformation presented in

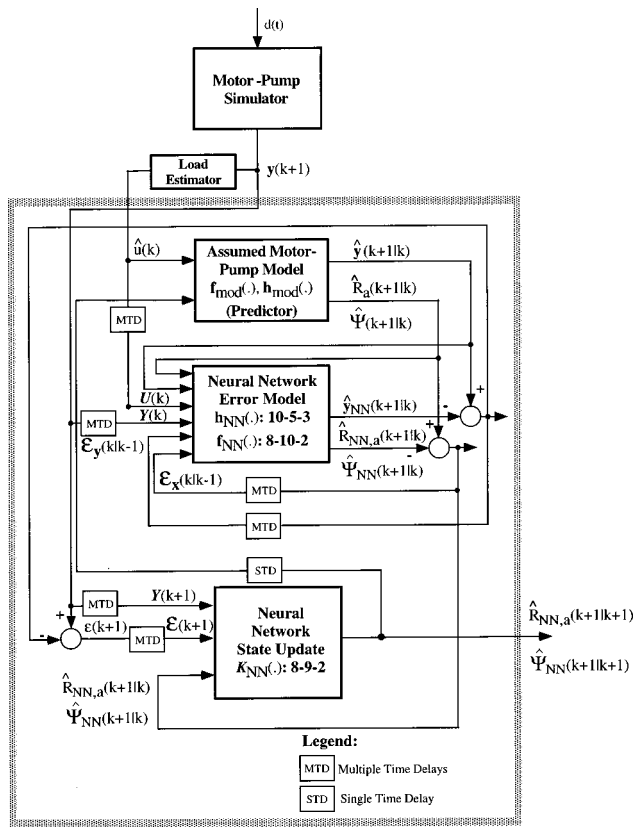


Fig. 2 Motor-pump system armature resistance and flux linkage filter block diagram

many digital control texts to Eqs. (18) through (21) [31]. In view of the assumed motor measurements $\mathbf{h}_{\text{mod}} = [1, 0, 1, 1]^T$.

4.2 Development of the Neural State Filter. To design the neural filter the motor-pump assumed model, $\mathbf{f}_{\text{mod}}(\cdot)$ and $\mathbf{h}_{\text{mod}}(\cdot)$, is used to generate the training set. Zero-mean, white Gaussian process and measurement noise is also added to the simulator such that the measured variables exhibit SNR of 3.0 and 0.5 for low and high noise experiments, respectively. Testing of the filter is performed using the motor-pump simulator, given by Eqs. (18) through (21).

In this study, Ψ is modeled as a slowly varying parameter that is “constant” for a number of transients, and yet acquiring a new value for different set of transients. A training set is constructed consisting of sinusoidal and ramp signals for $R_a(t)$, and different constant signals for Ψ . The training set consists of 10,000 samples (7500 samples for estimation and 2500 samples for the validation or out-of-sample testing used to stop training). The test set consists of 2000 samples of sinusoidal and ramp signals with characteristics different than those used in the training set. A block diagram for this filter is depicted in Fig. 2.

4.2.1 Error Model Architectures. The $\mathbf{f}_{NN}(\cdot)$ and $\mathbf{h}_{NN}(\cdot)$ EM predictors are chosen to be three-layer FMLP dynamic networks with 8 and 10 nodes in the input layer, respectively. The inputs to $\mathbf{f}_{NN}(\cdot)$ are the two state predictions from the assumed model, the two predicted state errors, the three measured outputs and the single measured system input. The inputs to $\mathbf{h}_{NN}(\cdot)$ are the three output predictions from the assumed model, the three predicted output errors, the three measured outputs and the single measured system input. The number of nodes in the third layer of the predictors is set to two and three, corresponding to the two predicted states and the three predicted system outputs, respectively. The error models are then trained using the training set and

Table 1 Motor-pump system armature resistance and flux linkage filter test set errors

State Filter		E_{NMSE}		$\overline{\epsilon_{\text{act}}(k)}$		$\overline{\epsilon_{\text{mod}}(k)}$	
		\hat{R}_a	$\hat{\Psi}$	\hat{R}_a	$\hat{\Psi}$	\hat{R}_a	$\hat{\Psi}$
Neural Filter w/o	Low Noise ¹	0.12%	0.10%	1.50%	2.30%	0.01%	0.02%
	High Noise ²	0.22%	0.18%	4.30%	2.60%	0.02%	0.04%
Neural Filter w/	Low Noise	0.01%	0.01%	0.08%	0.07%	0.06%	0.08%
	High Noise	0.02%	0.02%	0.09%	0.06%	0.08%	0.10%
Extended	Low Noise	0.11%	0.12%	1.45%	2.40%	0.02%	0.03%
	High Noise	0.25%	0.20%	4.20%	2.50%	0.03%	0.04%
Kalman Filter							

¹ Low Noise: $SNR = 3.0$

² High Noise: $SNR = 0.5$

varying number of hidden nodes. The EM networks with the best architecture are determined by trial-and-error to be 8–10–2 and 10–5–3, with test errors of $E_{NMSE} = 3.9\%$ and $E_{NMSE} = 1.0\%$, respectively.

4.2.2 State Update Architecture. The state update, $\mathcal{K}_{NN}(\cdot)$, is chosen to be a three-layer FMLP static network with 8 nodes in the first layer, corresponding to the inputs, and 2 nodes in the third layer, corresponding to the outputs. The inputs to the first layer are the three measured outputs, the three residuals, and the two corrected state predictions. The two outputs are the two estimated states. The state update network is then trained with the training set and with varying number of hidden nodes. The targets are $R_{ma}(k)$ and $\Psi_{ma}(k)$, as determined by the assumed motor-pump model. An 8–9–2 FMLP architecture is determined by trial-and-error to be the best for the state filter update network. The test error for this architecture is $E_{NMSE} = 0.5\%$.

4.3 Development of the Extended Kalman Filter. For comparison purposes, an EKF is designed for estimating the armature resistance and the flux linkage simultaneously. The standard discrete-time formulation of the EKF is used. For a fair comparison with the neural state filters, the previously presented motor-pump predictor and its linearized version, are used in the EKF design. Specifically, in the *Prediction Step* of the EKF, the predictor equations (24) are used, whereas the same equations are linearized for use in the covariance matrix calculations. In the *Update Step* of the EKF, the linearized form of the predictor 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 made in this simulation study. An attempt was also made to develop a linearized KF for the problem at hand. The results were not satisfactory because on a number of simulation studies the linearized filter did not converge.

4.4 Comparison of the Neural Filters and the EKF. The comparison of the neural filters and the EKF for the dual armature resistance and flux linkage study are shown in Table 1, and in Figs. 3 and 4. The simulations presented are for the test set previously discussed.

Figures 3 and 4 depict the state estimates as functions of time for the neural filter with and without on-line learning, and for the EKF. All filtering results are compared with the simulated states. The EKF exhibits response similar to the neural filter without on-line learning, while on-line learning improves the state filtering results. In fact, for most of the simulations performed using the neural filter with on-line learning, the mean state estimation error with respect to the simulated states approaches zero.

There are some discrepancies in the transients, but the overall responses are comparable, as witnessed by the results of Table 1. This table shows the averages of the state estimation errors with respect to the simulated states and the assumed model states. The most important observation is the value of the average estimation error with respect to the simulated states. Both the neural filter without on-line learning and the EKF show considerable error in

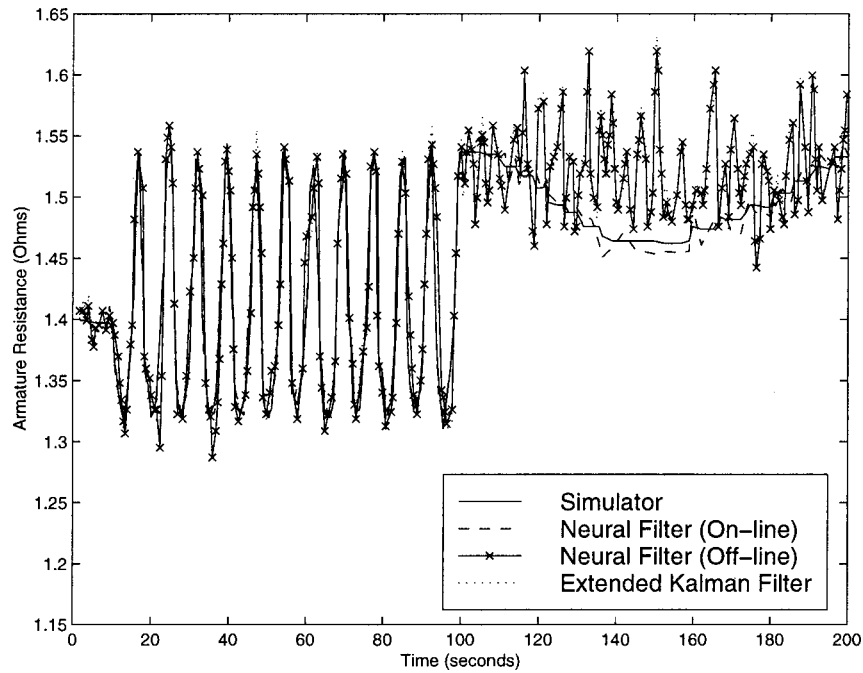


Fig. 3 Motor-pump system armature resistance and flux linkage filter response using the test data set as inputs (high noise environment)

this category because of the uncertainty in the assumed model used in developing them. At the same time, the estimation errors of these two filters with respect to the assumed model are quite low. However, further on-line learning reduces the estimation error with respect to the simulated states, at the expense of the estimation error with respect to the assumed model states. This is expected because the filter with on-line learning drifts away from the assumed model used in the off-line learning. Finally, one should expect that, in general, the estimation results with high

noise to be worse than the corresponding results with low noise. The flux linkage shows an exception to this expectation, but the error difference is very small.

5 The Case Study of a Complex Process System

In this section, an adaptive neural state filter is developed for estimating the UTSG riser void fraction, $\alpha_R(t)$.

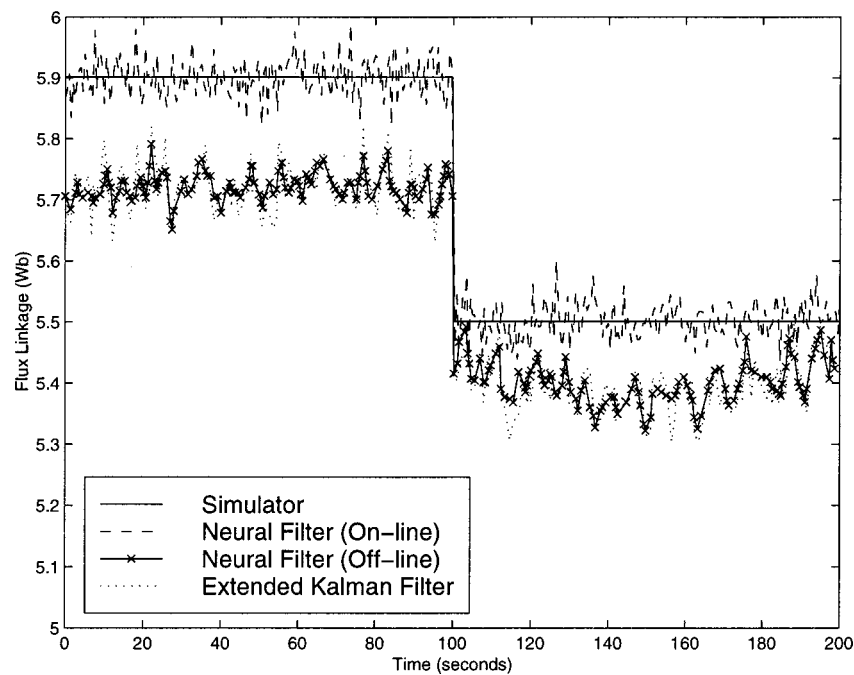


Fig. 4 Motor-pump system armature resistance and flux linkage filter response using the test data set as inputs (high noise environment)

5.1 Process System Description. A UTSG simulator developed by Choi [32] is adopted for the purpose of this study. Because the open-loop system is unstable, a stabilizing controller is required to allow system operation. The controller structure used with the above UTSG simulator is that proposed by Menon and Parlos [33]. The UTSG has one control input, $W_{fw}(t)$, and five disturbance inputs, namely, $T_{hi}(t)$, $W_{pr}(t)$, $W_{st}(t)$, $T_{fw}(t)$, and $P_{pr}(t)$. The only measured output of the UTSG used in this study is $L_w(t)$.

The UTSG can be represented in the following nonlinear state space representation:

$$\frac{d\mathbf{x}_{UTSG}(t)}{dt} = \mathbf{f}_{UTSG}(\mathbf{x}_{UTSG}(t), u_{UTSG}(t), \mathbf{w}_{UTSG}(t)), \quad (25)$$

where the state vector includes both the primary and secondary side states. The system of these nonlinear ordinary differential equations is solved to advance the transient simulation. The measured UTSG output is expressed in terms of its system states, as follows:

$$y_{UTSG}(t) = L_w(t) = h_{UTSG}(\mathbf{x}_{UTSG}(t)). \quad (26)$$

5.2 Development of the Riser Void Fraction Neural State Filter. The training and test sets used in the development and testing of the neural networks involved in the state filter are a collection of UTSG simulator and assumed model data, corresponding to step and ramp changes in operating power level. Only three of the five disturbances are used, along with the single controlled input of the UTSG. These are collectively referred to by the four-dimensional vector $\mathbf{u}(k)$. In addition, the simulator contains white, Gaussian process and measurement noise, such that the measured input and output variables exhibit SNR of 3.0 and 0.5 for low and high noise experiments, respectively.

The state filtering method is now applied for estimating the riser void fraction of the UTSG. In this filter, a scheduled (or piecewise linear) UTSG model is used as the filter model, $f_{mod}^{UTSG}(\cdot)$ and $h_{mod}^{UTSG}(\cdot)$, as further described in [33]. The training set used in the state filter development comprises of data collected from the scheduled UTSG model subjected to different step and ramp changes in the operating power level. The state values for the riser void fraction are also obtained from the scheduled UTSG model. The test set includes 2 individual tests; a ramp increase of 0.6% per minute from 5% to 10% of full power, and a step increase 15% to 18% of full power. Test sets with both low and high noise, as defined in the previous paragraph, are generated. The neural filter setup used in this case study is shown in Fig. 5.

5.2.1 Error Model Architectures. The $f_{NN}(\cdot)$ and $h_{NN}(\cdot)$ EM predictor networks are chosen to be three-layer FMLP dynamic networks with 7 nodes in the first layer, and 1 node in the third layer. The inputs to $f_{NN}(\cdot)$ are the state prediction from the assumed model, the predicted state error, the measured output and the four measured system inputs, including the disturbances. The inputs to $h_{NN}(\cdot)$ are the output prediction from the assumed model, the predicted output error, the measured output and the four measured system inputs, including the disturbances. The number of nodes in the third layer of the predictors is set to one, corresponding to the predicted state and the predicted output, respectively. The error models are trained using the training set and a varying number of hidden nodes. The EM networks with the best architecture are determined by trial-and-error to be 7-5-1 and 7-4-1, with test errors of $E_{NMSE} = 1.98\%$ and $E_{NMSE} = 1.33\%$, respectively.

5.2.2 State Update Architecture. The state update, $\mathcal{K}_{NN}(\cdot)$, is chosen to be a three-layer FMLP static network with 3 nodes in the first layer, corresponding to the inputs, and 1 nodes in the third layer, corresponding to the outputs. The inputs to the first layer are the measured output, the residual, and the corrected state prediction. The output node is the estimated state. The state update

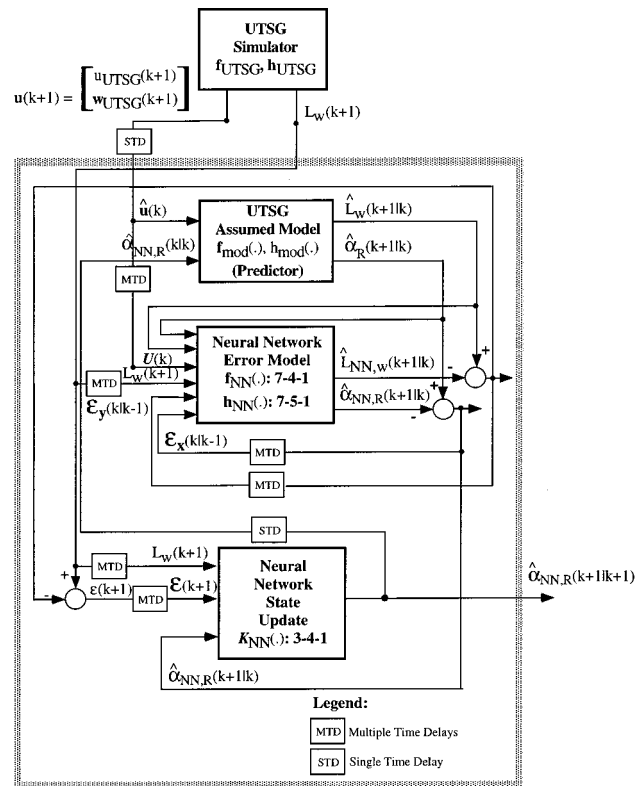


Fig. 5 Block diagram of the UTSG riser void fraction filter

network is then trained with the training set and with a varying number of hidden nodes. The target in training is the state $\alpha_{mR}(k)$, obtained from the UTSG scheduled model. The best FMLP network is found by trial-and-error to be a 3-4-1 network, with the average test error, E_{NMSE} , computed as 0.11%.

5.3 Development of the Extended Kalman Filter. For comparison purposes, an attempt has been made to develop an EKF for the UTSG case study. The first difficulty in this attempt has been the unavailability of analytical nonlinear or linearized models in state-space form for use in the EKF equations. In particular, the vector functions ($\mathbf{f}(\cdot)$, $\mathbf{h}(\cdot)$) and their Jacobians, both used in the EKF algorithm, are related through the differentiation operator, necessitating the availability of an analytic form. Furthermore, the same vector functions ($\mathbf{f}(\cdot)$, $\mathbf{h}(\cdot)$) used in the EKF, are assumed to be a perfect representation of the system. In this study, Choi's simulator [32] is used for testing the effectiveness of all the state filters. As previously described, the simulator is numerically linearized at many steady-state operating points and the resulting linearized models are scheduled (or fitted) to obtain a piecewise linear model [33]. The scheduled model is used in the EKF design instead of an analytic nonlinear model. Furthermore, the numerically linearized individual models are used in the EKF error covariance and filter gain calculations.

After overcoming these serious modeling difficulties, an EKF is developed for the UTSG riser void fraction. Filter validation is attempted using ramp and step changes in the UTSG operating power level. 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. Traditionally, EKFs are known to have convergence problems even when analytical nonlinear models of the underlying processes are available. Usually, one would attempt a linearized KF to reduce the impact of convergence problems. In the UTSG case study, a linearized KF did not achieve convergence either. The poor performance of the EKF for the UTSG riser void problem can be attributed to two

Table 2 UTSG process riser void fraction filter test set errors

Neural State Filter		E_{NMSE}	$\epsilon_{act}(k)$	$\epsilon_{mod}(k)$	
Filter without On-line Learning	Low Noise ³	Ramp Input	0.56%	0.38%	0.34%
		Step Input	0.05%	0.18%	0.12%
	High Noise ⁴	Ramp Input	4.50%	-2.60%	-2.48%
		Step Input	0.11%	0.26%	0.24%
Filter with On-line Learning	Low Noise	Ramp Input	0.08%	0.09%	0.07%
		Step Input	0.01%	0.02%	0.01%
	High Noise	Ramp Input	1.10%	0.65%	0.52%
		Step Input	0.01%	0.05%	0.04%

³ Low Noise: SNR = 3.0

⁴ High Noise: SNR = 0.5

reasons: (1) the lack of an analytical nonlinear model for use in obtaining a linearized model, and, (2) the introduction of structural modeling differences in the scheduled model used in the EKF design, compared to the simulator used in testing the EKF.

5.4 Comparison of the Neural Filters and the EKF. The comparison of the void fraction neural filters and the EKF are shown in Table 2, and in Figs. 6 and 7. The simulations presented in these figures are for the test set previously discussed.

The estimated riser void fraction for the ramp input of the test data set is shown in Fig. 6. The estimated riser void fraction for the step input of the test data set is shown in Fig. 7. These figures depict the state estimates as a function of time for the neural filter with and without on-line learning, and for the EKF. All filtering results are compared with the simulated state. The EKF estimates diverge from the simulated state values, while the neural filter without on-line learning have some steady-state errors. The neural state filtering results obtained in these tests indicate that the riser void fraction filter performs quite well. The filter's performance is further enhanced by allowing on-line learning. Considering the complexity of the underlying process, the results from both neural filters are quite encouraging. However, on-line learning significantly improves filter performance. Again, for most of the simulations performed using the neural filter with on-line learning the mean state estimation error approaches zero.

There are some discrepancies in the two convergent neural filter transients, but the overall responses are satisfactory, as witnessed by the results of Table 2. This table shows the averages of the state estimation errors with respect to the simulated state and the assumed model state. The most important observation is the value of the average estimation error with respect to the simulated state. The neural filter without on-line learning shows some error in this category because of the uncertainty in the assumed model used in developing it. At the same time, the estimation error of this filter with respect to the assumed model is lower. For the filter with on-line learning the estimation error with respect to the simulated state is further reduced, while also further reducing the estimation error with respect to the assumed model. This is in contrast to the results of the previous case study, though we do not have a reasonable explanation. Finally, one should note that, in general, the estimation results with high noise are worse than the corresponding results with low noise.

6 Summary and Conclusions

A practical algorithm is presented for adaptive state filtering in nonlinear dynamic systems using neural networks. The algorithm involves a prediction step and an update step, similar to the EKF framework. In the proposed formulation an assumed, approximate system model is considered available, though not necessarily in analytic form. Further, an EM is empirically identified to compensate for the unmodeled dynamics. A training procedure is presented for the filter development, consisting of an off-line and an on-line phase.

The adaptive state filtering algorithm is used to estimate the armature resistance and the magnetic flux linkage of a motor-pump system simultaneously. The armature resistance is considered a system state which cannot be directly measured on-line. This variable has the potential to indicate developing faults in DC motors. During load variations, the magnetic flux linkage is a parameter which varies slowly with time, compared to the armature resistance. Therefore, it is considered as a constant parameter. The resulting neural filter provides quite accurate state estimates. Furthermore, it is observed that the accuracy of the filter is not affected by the high noise environment considered. In all cases

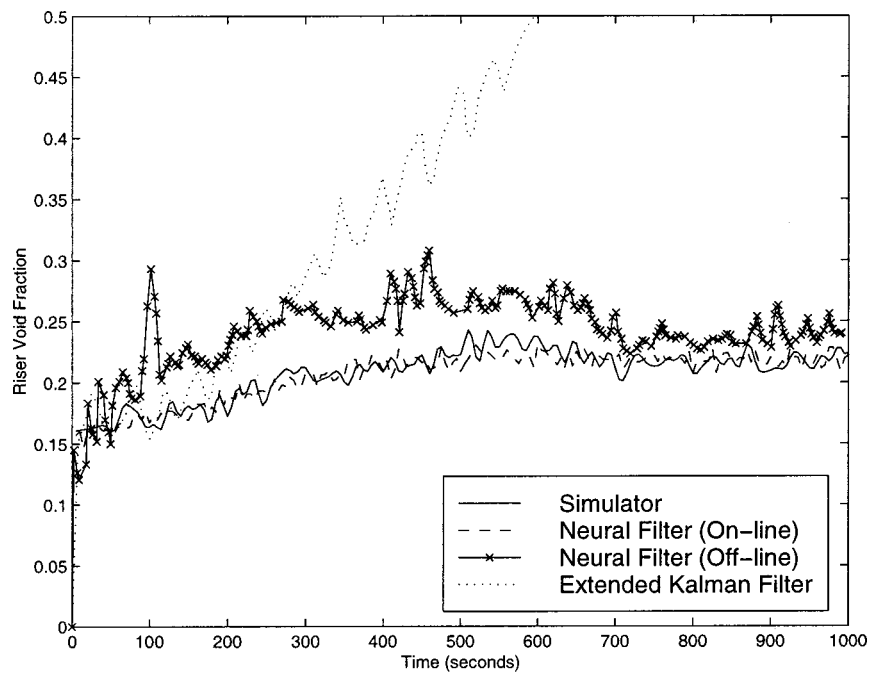


Fig. 6 UTSG process riser void fraction filter response using the ramp input of the test data set (high noise environment)

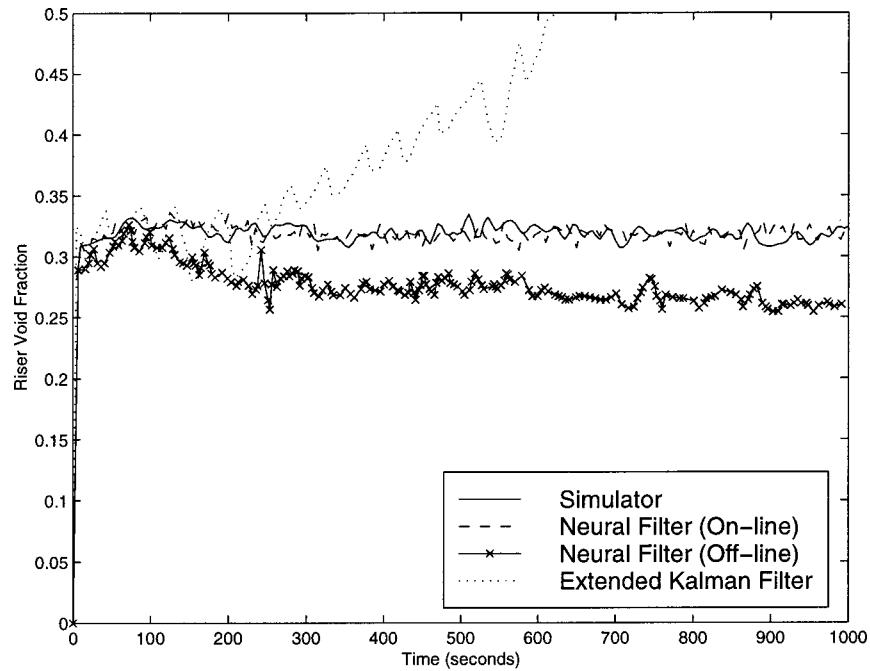


Fig. 7 UTSG process riser void fraction filter response using the step input of the test data set (high noise environment)

examined, use of the on-line learning has further improved the state filter accuracy. These observations indicate the potential for using such filters in applications involving condition monitoring and fault diagnosis.

The second case study presented is that of a UTSG. The UTSG system is open-loop unstable. Thus for the transients considered, a controller is used to stabilize it. An adaptive neural filter is developed to estimate the riser void fraction, a state of the UTSG process system. An EKF is also developed for this parameter. Furthermore, adaptive estimation results with and without on-line learning are presented. The EKF does not converge to a steady-state value. On the contrary the adaptive neural filter converges quite rapidly. Even without on-line learning the neural estimation results are acceptable compared to the EKF, which diverges quite rapidly. However, with the addition of on-line learning there is significant improvement in the neural state estimates, resulting in estimation errors with respect to simulated states of less than 1%. The test results for this adaptive filter affirm that the proposed algorithm can be applied to complex filtering problems with constant or slowly varying parameters.

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 fact couples the state estimation accuracy to the accuracy of the assumed system model used in filter design. On-line adaptation further improves the state estimates, though a proper on-line learning algorithm must be used. The on-line learning results further demonstrate that to the extent possible the state estimation error has been decoupled from the accuracy of the assumed model used in the estimators. Finally, our experience indicates that as the uncertainty and complexity of the state filtering problem increase, the benefit of using the proposed algorithm becomes more apparent. It is only then that complex state filters, such as the one proposed in this study, can be justified.

Acknowledgments

The authors greatly appreciate the financial support provided to Texas A&M University by the U.S. Department of Energy under grant DE-FG07-89ER12893 and the NASA Johnson Space Center

under grant NAG#9-347. Dr. Atiya would also like to acknowledge the support of NSF's Engineering Research Center at CalTech.

Nomenclature

Acronyms

KF, EKF	= Kalman and Extended Kalman Filter
UTSG	= U-Tube Steam Generator
TF, GF	= Teacher Forcing, Global Feedback
SSP, MSP	= Single-step-ahead Prediction, Multi-step-ahead Prediction
EM	= "Error Model"
FMLP, RMLP	= Feedforward and Recurrent Multilayer Perceptron
FIR, IIR	= Finite and Infinite Impulse Response
NMSE	= Normalized Mean Squared Error
SNR	= Signal-to-Noise Ratio

Mathematical Symbols

t, k	= continuous time, discrete time
$\mathbf{x}(t), \mathbf{y}(t), \mathbf{u}(t)$	= simulated system state, output and input vector
$\mathbf{f}(\cdot), \mathbf{h}(\cdot)$	= simulated nonlinear system dynamics
$\mathbf{z}(t), \mathbf{v}(t)$	= simulated process and sensor noise vectors
$\hat{\mathbf{p}}(k+1 k)$	= estimate of a variable \mathbf{p} at $k+1$ using measurements at k
$\mathbf{f}_{\text{mod}}(\cdot), \mathbf{h}_{\text{mod}}(\cdot)$	= modeled nonlinear system dynamics
$\hat{\mathbf{x}}(k+1 k), \hat{\mathbf{y}}(k+1 k)$	= state and output predictions by system model
$\mathbf{f}_{NN}(\cdot), \mathbf{h}_{NN}(\cdot)$	= neural network state and output EMs

$\hat{\mathbf{x}}_{NN}(k+1|k), \hat{\mathbf{y}}_{NN}(k+1|k)$ = state and output predictions by neural EM
 $\hat{\mathbf{x}}_{NN}(k|k)$ = adaptive state filter estimate
 $\mathbf{U}(k), \mathbf{Y}(k)$ = vectors containing measured past system inputs and outputs
 $\boldsymbol{\epsilon}(k), \boldsymbol{\epsilon}_x(k+1|k), \boldsymbol{\epsilon}_y(k+1|k)$ = innovations, and state/output EM errors
 $\mathcal{E}(k), \mathcal{E}_x(k+1|k), \mathcal{E}_y(k+1|k)$ = vectors of past innovations, and past state/output EM errors
 n_y, n_u, n_e = number of past output, input and innovations delays
 n_e^x, n_e^y = number of past EM error delays
 $\mathcal{K}_{NN}(\cdot)$ = neural network adaptive state filter gain
 $\mathbf{x}_{\text{target}}(k), \mathbf{y}_{\text{target}}(k)$ = state and output measurements used in training set
 $E(\cdot), W_s, \mathbf{w}$ = neural network objective function, training window and weights
 NP, K = number of training samples
 \mathcal{L}, K = network output index, number of network outputs
 $\boldsymbol{\epsilon}_{\text{act}}(k), \boldsymbol{\epsilon}_{\text{mod}}(k)$ = estimation errors relative to simulated and assumed model
 $\overline{\boldsymbol{\epsilon}_{\text{act}}(k)}, \overline{\boldsymbol{\epsilon}_{\text{mod}}(k)}$ = average value of estimation errors
 Ψ, L_a, K_v = motor flux linkage, inductance, and motor constant
 $V_a(t), I_a(t), \omega(t)$ = motor armature voltage, current and angular velocity
 $R_a(t), R_a^o, \alpha$ = motor armature resistance (AR), nominal AR, load constant
 θ_{mp} = motor-pump combined inertia
 c_{mp0}, c_{mp1} = Motor-pump static and dynamic friction coefficients
 $h_2, h_{nn}, h_{rr}(t)$ = Pump constant, valve/piping flow resistance, pump parameter
 $\dot{M}(t), \hat{T}_L(t)$ = Pump flow-rate and load
 K_{mM}, k = Empirical pump coefficients in motor-pump model
 $\hat{I}_a(k|k), \hat{I}_a(k+1|k)$ = Motor armature current estimates from assumed model
 $\hat{\omega}(k|k), \hat{\omega}(k+1|k)$ = Motor angular velocity estimates from assumed model
 $\hat{R}_a(k|k), \hat{R}_a(k+1|k)$ = Motor armature resistance estimates from assumed model
 $\hat{M}(k+1|k)$ = Pump flow-rate estimate from assumed model
 $f_{\text{mod}}^{mp1}(\cdot), f_{\text{mod}}^{mp2}(\cdot), f_{\text{mod}}^{mp3}(\cdot), f_{\text{mod}}^{mp4}(\cdot)$ = Discrete-time equivalents of the assumed motor-pump model
 $R_m(k), \Psi_m(k)$ = Target values of armature resistance and flux linkage
 $\hat{R}_{NN,a}(k|k), \hat{\Psi}_{NN}(k|k)$ = Filtered values of armature resistance and flux linkage
 $W_{fw}(t), T_{fw}(t)$ = UTSG feed-water flow-rate and temperature
 $T_{hl}(t)$ = UTSG hot-leg temperature
 $W_{pr}(t), W_{sr}(t)$ = UTSG primary and secondary side flow-rates
 $P_{pr}(t)$ = UTSG primary side pressure
 $L_w(t)$ = UTSG downcomer water level
 $\alpha_R(t)$ = UTSG void fraction

$\mathbf{x}_{\text{UTSG}}, \mathbf{y}_{\text{UTSG}}$,
 $\mathbf{u}_{\text{UTSG}}, \mathbf{w}_{\text{UTSG}}$ = UTSG state, output, input and disturbance vectors
 $\mathbf{f}_{\text{UTSG}}(\cdot), h_{\text{UTSG}}(\cdot)$ = UTSG simulated nonlinear dynamics
 $\mathbf{f}_{\text{mod}}^{\text{UTSG}}(\cdot), h_{\text{mod}}^{\text{UTSG}}(\cdot)$ = UTSG assumed nonlinear dynamics
 $\alpha_{m,R}(k)$ = target value for UTSG void fraction
 $\hat{\alpha}_R(k|k), \hat{\alpha}_R(k+1|k)$ = UTSG void fraction estimates from assumed model
 $\hat{\alpha}_{NN,R}(k|k)$ = Filtered value of UTSG void fraction

References

- [1] Kalman, R. E., and Bucy, R. S., 1961, "New Results in Linear Filtering and Prediction Theory," *ASME J. Basic Eng.*, **83**, pp. 95–107.
- [2] Gelb, A., 1974, *Applied Optimal Estimation*, MIT Press, Cambridge, MA.
- [3] Jonsson, G., and Palsson, O. P., 1994, "An Application of Extended Kalman Filtering to Heat Exchanger Models," *ASME J. Dyn. Syst., Meas., Control*, **116**, pp. 257–264.
- [4] Haykin, S., 1999, *Neural Networks: A Comprehensive Foundation*, 2nd Edition, Prentice-Hall, Piscataway, NJ.
- [5] Lo, J. T-H., 1994, "Synthetic Approach to Optimal Filtering," *IEEE Trans. Neural Netw.*, **5**(5) Sept., pp. 803–811.
- [6] Elanayar, S., and Shin, Y. C., 1994, "Radial Basis Function Neural Network for Approximation and Estimation of Nonlinear Stochastic Dynamic Systems," *IEEE Trans. Neural Netw.*, **5**(4) July, pp. 594–603.
- [7] Annaswamy, A. M., and Yu, S. H., 1996, " θ -Adaptive Neural Networks: A New Approach to Parameter Estimation," *IEEE Trans. Neural Netw.*, **7**(4), pp. 594–603.
- [8] Parisini, T., Alessandri, A., Maggiore, M., and Zoppoli, R., 1997, "On Convergence of Neural Approximate Nonlinear State Estimators," *Proceedings of the 1997 American Control Conference*, Vol. 3, June, pp. 1819–1822.
- [9] Alessandri, A., Parisini, T., and Zoppoli, R., 1997, "Neural Approximators for Nonlinear Finite-Memory State Estimation," *Int. J. Control*, **67**(2), pp. 275–301.
- [10] Haykin, S., Yee, P., and Derbez, E., 1997, "Optimum Nonlinear Filtering," *IEEE Trans. Neural Netw.*, **45**(11) Nov., pp. 2774–2786.
- [11] Zhu, R., Chai, T., and Shao, C., 1997, "Robust Nonlinear Adaptive Observer Design using Dynamic Recurrent Neural Networks," *Proceedings of the 1997 American Control Conference*, Vol. 2, June, pp. 1096–1100.
- [12] Habtom, R., and Litz, L., 1997, "Estimation of Unmeasured Inputs using Recurrent Neural Networks and the Extended Kalman Filter," *International Conference on Neural Networks*, Vol. 4, pp. 2067–2071.
- [13] Dong, X., Qui, L., and Wang, Z., 1997, "Neural Networks-based Nonlinear Adaptive Filters and On-line Fault Detection," *Control and Decision*, **12**(1) Jan., pp. 78–87.
- [14] Lei, J., Guangdong, H., and Jiang, J. P., 1997, "The State Estimation of the CSTR System Based on a Recurrent Neural Network Trained by HGAs," *International Conference on Neural Networks*, Vol. 2, pp. 779–782.
- [15] Schenker, B., and Agarwal, M., 1998, "Predictive Control of a Bench-Scale Chemical Reactor Based on Neural-Network Models," *IEEE Trans. Control Syst. Technol.*, **6**(3) May, pp. 388–400.
- [16] Stubberud, S. C., and Owen, M., 1998, "Targeted On-line Modeling for an Extended Kalman Filter Using Artificial Neural Networks," *Proceedings of the 1998 American Control Conference*, Vol. 3, June, pp. 1852–1856.
- [17] Stubberud, S. C., Owen, M., and Lobbia, R. N., 1998, "Adaptive Extended Kalman Filter Using Artificial Neural Networks," *International Journal of Smart Engineering System Design*, **1**(3), pp. 207–221.
- [18] Durovic, Z., and Kovacevic, B., 1998, "Adaptive Filtering using Neural Networks Approach," *Proceedings of the Mediterranean Electrotechnical Conference*, Vol. 1, May, pp. 499–503.
- [19] Menon, S. K., Parlos, A. G., and Atiya, A. F., 2000, "Nonlinear State Filtering for Fault Diagnosis and Prognosis in Complex Systems Using Recurrent Neural Networks," *4th Symposium on Fault Detection, Supervision and Safety for Technical Processes*, IFAC SAFEPROCESS 2000, June.
- [20] Parlos, A. G., Menon, S. K., and Atiya, A. F., 1999, "Adaptive State Estimation Using Dynamic Recurrent Neural Networks," *Proceedings of the International Joint Conference on Neural Networks*, June.
- [21] Grewal, M. S., and Andrews, A. P., 1993, *Kalman Filtering: Theory and Practice*, Prentice-Hall, Upper Saddle River, NJ.
- [22] Ljung, L., 1999, *System Identification: Theory for the User*, 2nd Edition, Prentice-Hall, Upper Saddle River, NJ.
- [23] Narendra, K. S., and Parthasarathy, K., 1990, "Identification and Control of Dynamic System Using Neural Networks," *IEEE Trans. Neural Netw.*, **1**, pp. 4–27.
- [24] Barron, A. R., 1994, "Approximation and Estimation Bounds for Artificial Neural Networks," *Journal of Machine Learning*, **14**, pp. 115–133.
- [25] Parlos, A. G., Rais, O. T., and Atiya, A. F., 2000, "Multi-Step-Ahead Prediction in Complex Systems Using Dynamic Recurrent Neural Networks," *Neural Networks*, **13**(4–5), pp. 765–786.

- [26] Parlos, A. G., Chong, K. T., and Atiya, A., 1994, "Application of the Recurrent Multilayer Perceptron in Modeling Complex Process Dynamics," *IEEE Trans. Neural Netw.*, **5**, pp. 255–266.
- [27] Williams, R., and Zipser, D., 1989, "A learning algorithm for continually running fully recurrent neural networks," *Neural Comput.*, **1**, pp. 270–280.
- [28] Atiya, A., and Parlos, A., 2000, "New Results on Recurrent Network Training: Unifying the Algorithms and Accelerating Convergence," *IEEE Trans. Neural Netw.*, **11**, pp. 697–709.
- [29] Geiger, G., 1984, "Fault Identification of a Motor-Pump System using Parameter Estimation and Pattern Classification," *IFAC 9th Triennial World Congress*, Dec.
- [30] Isermann, R., 1985, "Process Fault Diagnosis with Parameter Estimation Methods," *IFAC Digital Computer Applications to Process Control*, Dec., pp. 51–60.
- [31] Franklin, G. F., Powell, J. D., and Workman, M., 1998, *Digital Control of Dynamic Systems, 3rd Edition*, Addison Wesley Longman, Menlo Park, CA.
- [32] Choi, J. I., 1987, *Nonlinear Digital Computer Control for the Steam Generator System in a Pressurized Water Reactor Plant*, PhD thesis, MIT, Department of Nuclear Engineering.
- [33] Menon, S. K., and Parlos, A. G., 1992, "Gain-Scheduled Nonlinear Control of U-Tube Steam Generator Water Level," *Nuclear Science and Engineering*, **III**(3), pp. 294–308.