Recursive Neural Filters and Dynamical Range Transformers

JAMES T. LO AND LEI YU

Invited Paper

A recursive neural filter employs a recursive neural network to process a measurement process to estimate a signal process. This paper reviews and presents new results on recursive neural filters through introducing those based on a multilayer perceptron with output feedbacks and proposing dynamical range reducers and extenders for applications where the range of the measurement or signal process expands in time or is too large for the recurrent neural network to handle for the filtering resolution required.

As opposed to the conventional analytic approach to deriving filter equations, recursive neural filters are synthesized from exemplary realizations of the signal and measurement processes, which are obtained by either computer simulations or actual experiments. No assumptions such as linear dynamics, Gaussian distribution, additive noise, and Markov property are required.

A properly trained recursive neural filter with a proper architecture carries the most "informative" statistics in its dynamical state and approximates the optimal performance to any accuracy. The recursive neural filter is a massively parallel algorithm ideal for real-time implementation.

Dynamical range reducers and extenders proposed are preprocessors and postprocessors of a recurrent neural network that, respectively, reduce the range of the exogenous input and extend that of the output of the recurrent neural network. While a dynamical range reducer reduces the exogenous input range by subtracting an estimate of the exogenous input from it, a dynamical range extender extends the output range by adding an estimate of the output to it.

Three types of dynamical range reducer by estimate subtraction and five types of dynamical range extender by estimate addition are provided, which have different levels of effectiveness and computational costs. Selection from the different types for an application is determined by the tradeoff between the effectiveness and computational cost.

Two types of dynamical range extender by estimate addition use an extended Kalman filter (EKF) to generate the added estimate. They can be viewed as methods of using a recursive neural filter only to make up the nonlinear part ignored by the EKF.

Keywords*—Dynamical range extender, dynamical range reducer, Kalman filter (KF), measurement, minimum variance, optimal filtering, recursive filtering, recursive neural filter, recursive neural network, signal, synthetic approach.*

Manuscript received February 18, 2003; revised October 28, 2003. This work was supported in part by the National Science Foundation under Grants ECS-0114619 and ECS-9707206, in part by the Army Research Office under Contract DAAD19-99-C-0031, and in part by the Rome Laboratory under Contract F30602-91-C0033.

James T. Lo is with the Department of Mathematics and Statistics, University of Maryland Baltimore County, Baltimore, MD 21250 USA (e-mail: jameslo@math.umbc.edu).

Lei Yu is with the Microsoft Corporation, Washington, DC 20015 USA (e-mail: leiyu@microsoft.com).

Digital Object Identifier 10.1109/JPROC.2003.823148

I. INTRODUCTION

In the standard formulation of the main problem in the modern theory of optimal filtering, the signal process and measurement process are described by the mathematical/statistical model

$$x_{n+1} = f(x_n, n) + G(x_n, n)d_n \tag{1}$$

$$y_n = h(x_n, n) + v_n \tag{2}$$

where x_n and y_n are N_x-dimensional and N_y-dimensional stochastic processes, respectively; x_1 is a Gaussian random vector, d_n and v_n are, respectively, N_d-dimensional and N_y-dimensional Gaussian noise processes with zero means; x_1, d_n, and v_n have given joint probability distributions; and $f(x_n, n), G(x_n, n)$, and $h(x_n, n)$ are known functions with such appropriate dimensions and properties that (1) and (2) describe the evolutions of the signal and measurement.

The problem of discrete-time optimal filtering is to design and make a discrete-time dynamical system that inputs y_n and outputs an estimate $\hat{x}_n$ of x_n at each time $n = 1, 2, \ldots, N$, which minimizes a given estimation error criterion. Here N is a positive integer or infinity. The dynamical system is called an optimal filter with respect to the given estimation error criterion. The dynamical state of the optimal filter at a time n_1 must carry the optimal conditional statistics given all the measurements y_n that have been received up to and including the time n_1 at the time so that at the next time $n_1 + 1$, the optimal filter will receive and process y_{n_1+1} using the optimal conditional statistics from n_1, and then produce the optimal estimate $\hat{x}_{n_1+1}$. The most widely used estimation error criterion is

the mean square error criterion $E[||x_n - \hat{x}_n||^2]$, where E and $||\cdot||$ denote the expectation and the Euclidean norm, respectively. The estimate $\hat{x}_n$ that minimizes this criterion is called the minimum-variance estimate.

If f and h are linear functions of x_n, and G does not depend on x_n, the model, (1) and (2), is called the linear-Gaussian model. A minimum-variance filter is the celebrated Kalman filter (KF) [1], [2] described in the first paper in this special issue by S. Haykin. In most real-world applications, however, the foregoing linearity conditions on f, h, and G are not satisfied and the extended KF (EKF) is used. At each time point, the EKF first linearizes f and G at the estimated value of x_n and linearizes h at the predicted value of x_{n+1}. Then the EKF uses the KF equations to update the estimated value of x_{n+1} and the predicted value of x_{n+2} for the new measurement y_{n+2}. By iterating the linearization and estimation a certain number of times or until convergence at each time point, we have the so-called iterated EKF (IEKF). Since both the EKF and IEKF involve linearization, they are not optimal filters. In fact, when either the random driving term $G(x_{n+1}, n)d_n$ in (1) or the random measurement noise v_n in (2) has such large variances or covariances that the aforementioned estimated value and predicted value of the signal are not very close to the true signal, or when the functions f, G, and h are not very smooth, linearization may be a poor approximation and the EKF as well as IEKF may yield poor estimates or even diverge. The reader is referred to S. Haykin's paper in this special issue for more descriptions and references for KFs and EKFs.

This shortcomings of the EKF and IEKF have motivated an enormous amount of work on nonlinear filtering by the analytic approach for over 30 years [3]–[5]. (More references can be found in the papers by Haykin and Krishnamurthy and Elliott in this special issue.) Starting with a mathematical model, the analytic approach searches for a solution or approximate solution consisting of equations that describe the structures or characterize the variables of the filter. In the process of searching, deductive reasoning is used and many assumptions are made to make some special cases analytically tractable. In fact, the KF was derived under the assumptions that f and h are linear in x_{n+1}, G does not depend on x_{n+1}, and d_n and v_n are Gaussian sequences. The general model, (1) and (2), contains such assumptions as the Markov property, Gaussian distribution, and additive measurement noise.

It is appropriate to mention here that many approximate nonlinear filters have been obtained, which can be viewed as predecessors of the recursive neural filters discussed in this paper. Among them are those derived by approximating non-Gaussian densities with Gaussian sums [6], [7], those obtained on group manifolds [8]–[11], and those obtained by functional series expansion [12]–[16]. These approximate filters are optimal for the given structure and approach the minimum variance filter as the number of terms increases. These are the properties shared by the recursive neural filters. However, those approximate filters often involve a great deal of computation and/or an excessive number of terms. The EKF and, to a much less extent, the IEKF remain as the standard filters for estimating stochastic signals in practice.

An alternative approach was proposed in 1992 [17], [18]. As opposed to the analytic approach, the proposed approach is synthetic in nature. Instead of deriving equations based on a mathematical model such as (1) and (2), the synthetic approach synthesizes data collected by actual experiments or computer simulations into a recursive filter that approximates a theoretical optimal filter in performance to any preselected degree of accuracy with respect to a given estimation criterion.

In the synthetic approach, a class of dynamical systems that are universal approximators of optimal filters is required. A most parsimonious class of this kind is multilayer perceptrons with interconnected neurons (MLPWINs) used in [17], [19]. The synthesis of a recursive filter is performed through training at least one MLPWIN and selecting one in consideration of the filtering performance versus the numbers of nodes and connections in the MLPWIN to optimize the cost-effectiveness. In training an MLPWIN, the adjustable weights (including biases) are determined essentially by minimizing or reducing a training criterion by the variation of them.

A training criterion, which is constructed with the training data, is a function of these adjustable weights of the MLPWIN under training. The training criterion does not have to be the standard minimum-variance criterion and can be defined to reflect an estimation error criterion suitable for the filtering needs and environment. Since we do not use a mathematical model of the signal and measurement processes to derive equations, such properties as the Markov property, Gaussian distribution, and additive noise are not required of the signal and measurement processes for the synthetic approach.

A properly trained MLPWIN with a proper architecture is a virtually optimal recursive filter, whose dynamical state at a time carries the most "informative" conditional statistics [17], [19] at the time for generating subsequent optimal estimates after the time. The massively parallel architecture of the MLPWIN makes it ideal for real-time filtering.

However, there are two shortcomings in using an MLPWIN as a recursive filter. First, if the mean or covariance of the initial signal (i.e., signal at time 0) is available, the information cannot be used by the MLPWIN, resulting in a longer and suboptimal transient filter state. Second, if the ranges of the signal and measurement expand over time, such as in financial time series prediction, satellite orbit determination, aircraft/ship navigation, and target tracking, or are large relative to the filtering resolution or accuracy required, the size of the MLPWIN and the training data set must be large. The larger the MLPWIN and the training data set are, the more difficult it is to train the MLPWIN on the training data set. Furthermore, the periods over which the training data is collected, by computer simulation or actual experiment, are necessarily of finite length. If the measurement and signal processes grow beyond these periods, the MLPWIN trained on the training data usually diverges after them.

The main purpose of this paper is to show how these two shortcomings can be eliminated. Our method of incorporating the mean or covariance of the initial signal is the use

of a recursive neural network (RNN) with output feedbacks such as a multilayer perceptron with output feedbacks (MLPWOF) instead of an MLPWIN. It is proven here that the minimum-variance filter can also be approximated to any accuracy by an MLPWOF. The mean or covariance of the initial signal can be loaded into the MLPWOF at the output feedback terminals before it starts filtering the measurements. MLPWOFs are numerically tested as recursive filters for the same signal and measurement processes as in [17] and [19]. Their filtering performances are shown to be similar to those of the neural filters with an MLPWIN and superior to those of the EKFs and IEKFs.

MLPWINs and MLPWOFs are two types of RNN widely used. There are other types. For instance, combinations of the two often work better. However, we will not discuss the RNN architecture selection further in this paper.

To eliminate the foregoing second difficulty, we propose the use of dynamical range transformers. There are two types of dynamical range transformer, called dynamical range reducers or extenders, which are preprocessors and postprocessors, respectively, of an RNN. A dynamical range reducer transforms dynamically a component of an exogenous input process (e.g., a measurement process) and sends the resulting process to the input terminals of an RNN so as to reduce the valid input range or approximation capability required of the RNN. On the other hand, a dynamical range extender transforms dynamically the output of an output node of an RNN so as to reduce the valid output range or approximation capability required of the RNN. The purpose of both the dynamical range reducer and extender is to ease the RNN size and training data requirements and thereby lessen the training difficulty. The fundamental range transforming requirement in using a dynamical range transformer (i.e., dynamical range reducer or extender) is the existence of a recursive filter comprising an RNN and the dynamical range transformer that approximates the optimal filter to any accuracy, provided that the RNN with the selected architecture is sufficiently large. A recursive filter comprising an RNN and dynamical range transformers is called a recursive neural filter.

A basic scheme to reduce the measurement range is to subtract an estimate of the current measurement from the current measurement, the estimate being based on measurements preceding the current measurement. A type of dynamical range reducer using this scheme are called range reducers by estimate subtraction. Three types of range reducer by estimate subtraction—namely, range reducers by differencing, range reducers by linear prediction, and range reducers by model-aided prediction—are introduced.

If the signal process is observable from the outputs of the dynamical range reducers in a recursive neural filter or if the mean and covariance of the initial signal are fixed, a recursive neural filter consisting of an MLPWIN or MLPWOF and dynamical range reducers by estimate subtraction satisfies the fundamental range transforming requirement. Otherwise, an RNN with output feedbacks such as an MLPWOF must be used, and a good estimate of the mean of the initial signal is needed to be loaded into the RNN in the recursive neural filter with dynamical range reducers by estimate subtraction at the beginning of its operation.

A basic scheme to extend the output range of an output node of an RNN is to add an estimate of the corresponding component of the signal to the actual output of the output node, the estimate being based on the measurements before the current time. A type of dynamical range extender using this scheme are called range extenders by estimate addition. Five types of range extender by estimate addition—namely, range extenders by accumulation, range extenders by Kalman filtering, range extenders by feedforward Kalman filtering, range extenders by linear prediction, and range extenders by feedforward linear prediction—are introduced.

A recursive neural filter with range extenders by feedforward Kalman filtering can be viewed as using an RNN to make up the loss caused by linearization in EKF. A requirement for this scheme to work is a reasonable performance of the EKF. Such a recursive neural filter can be used to upgrade existing EKFs. A theoretically better version is a recursive neural filter with range extenders by Kalman filtering, which allows interaction of a modified EKF and an RNN to enhance the performance of the modified EKF and to still make up the loss caused by linearization in the modified EKF. Even if no independent EKF works properly in an application, this recursive neural filter can still have a virtually optimal performance.

Most of the mentioned dynamical range transformers can be used in many other applications of recurrent neural networks than neural filtering.

The synthetic approach depends on successful training of the recursive neural filter. The error criteria used in training are nonconvex and the local-minimum problem has been a main obstacle in neural network training. Fortunately, an effective global minimization method has recently been developed for data fitting including neural network training. The reader is referred to [20]–[22] for detailed description, mathematical proof, and numerical confirmation of the method. The method requires evaluating derivatives of the training criterion. Pseudoprograms of the back-propagation through time (BPTT) for differentiating MLPWINs and MLPWOFs with various dynamical range transformers can be found in U.S. Patent 6 601 051, "Neural Systems with Range Reducers and/or Extenders," issued on July 29, 2003.[1]

The signal process to be estimated and the input process to be processed by a recursive neural filter are not defined in the same way as those of the conventional filtering theory. The signal and input processes for recursive neural filtering are defined in Section II. A general MLPWOF is described in detail in Section III. In Section IV, it is shown that the signal process can be used as the target process in training a recursive neural filter instead of the optimal estimate as would be expected in standard practice of neural network training.

It is proven in Section V that given a signal and measurement process, there exists an MLPWOF with only free feedbacks that approximates the conditional expectation of the signal given the past and present measurements to any accuracy. Three types of dynamical range reducer and five types of dynamical range extender are introduced in Sections VI and VII.

[1]Available: www.uspto.gov/patft/index.html

In Section VIII, four numerical examples show the filtering performances of recursive neural filters as compared with EKFs, a fifth compares a simple dynamical range extender with a static transformer for extending the output range of an RNN, and a sixth shows how the nonlinear error of an EKF caused by linearization is made up by a dynamical range extender by Kalman filtering.

Advantages of the synthetic approach to optimal filtering and the dynamical range transformers are summarized in Section IX. Some parts of this paper were reported in conferences [18], [23].

Since the synthetic approach to optimal filtering was proposed in [17]–[19] and [24], results on applications of various neural network paradigms to various filtering problems have been published. A neural filtering problem was reduced to a nonlinear programming one in [25]. A noise-free signal process was estimated using a feedforward or finit-memory neural network in [26]. Finite-memory filtering using neural networks was reported in [27] and [28]. References [29] and [30] presented an adaptive EKF and EKF implementations using neural networks. In [31], an adaptive nonlinear filter which mimics the KF's two-step prediction-update scheme was proposed using neural networks with all their weights adjusted online for adaptation.

Radial basis functions (RBFs) were used in [32] and [33] for nonlinear filtering. Reference [33] also contains a good survey of optimal filtering and a comparison of their RBF filters and the recursive neural filters in [18], [19], and [24].

Recently, particle filters and unscented KFs have attracted much attention [34]–[36]. They are also better than the EKF in performance. If a mathematical model of the signal and measurement process is given and has certain structures, a particle filter can also approximate optimal estimates to any accuracy provided enough particles are used. Particle filters do Monte Carlo online and, therefore, suffer from excessive computational requirements. In contrast, neural filter training can be viewed as Monte Carlo performed offline before the filter is deployed and, therefore, requires much less online computation. A detailed comparison of the neural filters, particle filters, and unscented filters is necessary. Nevertheless, some obvious advantages of the neural filters are stated in the conclusion of this paper.

II. Signal and Information Processes for Neural Filtering

The way to decide what signal process to use for recursive neural filtering is sometimes different from that for Kalman filtering. The former is much simpler and more straightforward than the latter. Let us illustrate the difference by an example. Suppose that we want to estimate a scalar-valued process x_{1n} that is the first component of a four-dimensional Markov process described by, for $n = 1, 2, \ldots, N-1$

$$\begin{bmatrix} x_{1(n+1)} \\ x_{2(n+1)} \\ x_{3(n+1)} \\ x_{4(n+1)} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_{1n} \\ x_{2n} \\ x_{3n} \\ x_{4n} \end{bmatrix} + \begin{bmatrix} \xi_{1n} \\ \xi_{2n} \\ \xi_{3n} \\ \xi_{4n} \end{bmatrix} \quad (3)$$

where $\xi_n = [\xi_{1n}, \xi_{2n}, \xi_{3n}, \xi_{4n}]^T$ is a standard four-dimensional white Gaussian sequence with mean 0 and covariance $E[\xi_{n_1}\xi_{n_2}^T] = \delta_{n_1 n_2} I_4$, I_4 being the 4×4 identity matrix, and where the initial state x_0 is a Gaussian random vector with mean 0 and covariance Σ_0 and statistically independent of ξ_n for all n. $\delta_{n_1 n_2}$ is the Kronecker delta, i.e., $\delta_{n_1 n_2} := 1$, if $n_1 = n_2$, and $\delta_{n_1 n_2} := 0$ otherwise. Assume that the measurement process y_n is described by $y_n = x_{4n} + \varepsilon_n$, where ε_n is a scalar valued (or equivalently one-dimensional vector-valued) white Gaussian sequence with mean 0 and variance 1 and statistically independent of the Markov process x_n.

In applying the KF to estimating x_{1n}, we need to include all the four components of $x_n, n = 1, 2, \ldots, N$, in the signal process for Kalman filtering. Notice that the first three components x_{1n}, x_{2n}, and x_{3n} of x_n form a Markov process by themselves. Nevertheless, this Markov process cannot be regarded as the signal process because its components do not enter the measurement process directly.

In fact, in applying the KF or the EKF, the signal process must be a Markov process that contain, as its components, all the processes (e.g., x_{1n} in the above example) being estimated, all the processes (e.g., x_{4n} in the above example) entering the measurement process directly, and all the processes (e.g., x_{2n} and x_{3n} in the above example) being required to make the signal process a Markov process.

However, in applying a recursive neural filter, the signal process consists only of the processes (e.g., x_{1n} in the above example) being estimated and any other statistics of it such as the conditional error covariance of its estimate, whether a mathematical in the form of (1) and (2) is available or not. In other words, the signal is the vector of all the quantities to be estimated by the recursive neural filter. This signal process for neural filtering is also denoted by $x_n, n = 1, 2, \ldots, N$, but may be different from the signal as defined in the conventional filtering theory. Bearing this in mind, the dual use of the symbol x_n should not cause any confusion, especially since the context is written to make it clear which process the symbol x_n refers to.

The inputs to a recursive neural filter should include all the information that is available and useful in estimating the signal. They comprise the entire measurement process. In addition, if the the signal or measurement process depends on a function of time, then this function of time should also be included. For instance, if the signal process is a controlled process, then the control function is a time function to be included. Furthermore, if a primary recursive neural filter is used to estimate only the signal process and an ancillary recursive neural filter is used to estimate the conditional error covariance, then the output of the former should be used as an input of the latter.

All the input processes of a recursive neural filter are called an information process. In the rest of the paper we will also denote an information process as well as a measurement process by y_n. Aided by the context, the reader is not expected to be confused by this abuse of the symbol.

III. Multilayer Perceptrons With Output Feedbacks

A typical multilayer perceptron (MLP) is illustrated in Fig. 1, which has two input nodes, three nodes in the first hidden layer, two nodes in the second hidden layer, and three

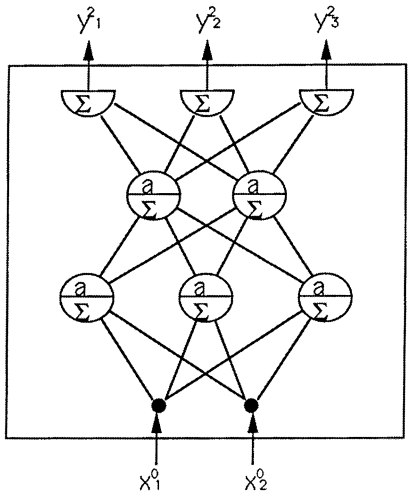

Fig. 1. An MLP.

output nodes. Let a general MLP have L layers, n_l nodes in layer l for $l = 0, \ldots, L$, where the n_0 nodes in layer 0 are the input terminals and the n_L nodes in layer L are the output nodes. Denoting the weighted sum in node i in layer l and the output of the same node, called its activation, be denoted by y_i^l and x_i^l, respectively. Then they satisfy the equations:

$$y_i^l = w_{i0}^l + \sum_j w_{ij}^l x_j^{l-1}$$
$$x_i^l = a\left(y_i^l\right)$$

for $l = 1, \ldots, L-1$, where a is the node activation function and w_{ij}^l is the weight on the connection from the node j in layer $l-1$ to node i in layer l. The inputs x_i^0 to the MLP are processed through the layers to produce the outputs y_i^L of the MLP, which are the weighted sums of the activations x_i^{L-1} of nodes in layer $L-1$.

The general MLPWOF is shown in Fig. 2, which is essentially an MLP with feedbacks from its output nodes after a unit-time delay represented by a solid square in the figure. We note here that in all the figures of this paper, the current time is denoted by t, and the time variables are enclosed in a pair of parentheses; although in the text, they are respectively denoted by n and expressed in the subscripts of other variables. It has $j + k$ output nodes, among which $\alpha_{1(n+1)}, \ldots, \alpha_{k(n+1)}$ are teacher-influenced outputs and $\beta_{1(n+1)}, \ldots, \beta_{j(n+1)}$ are free outputs at time $n + 1$. The MLP is trained to make only the teacher-influenced outputs mimic the desired outputs. All the free outputs and i of the teacher-influenced outputs $\alpha_{1(n+1)}, \ldots, \alpha_{i(n+1)}$ are delayed by one unit of time before being fed back to the input nodes $\beta_{nn}, \ldots, \beta_{jn}, \alpha_{1n}, \ldots, \alpha_{\text{in}}$, which are called free feedbacks and teacher-influenced feedbacks, respectively. These delayed feedbacks at time $n + 1$ constitute the dynamical state of the MLPWOF at the same time. The MLPWOF has m input terminals for the external inputs to the MLPWOF $\gamma_{1(n+1)}, \ldots, \gamma_{m(n+1)}$.

If an MLPWOF is to be used as a recursive neural filter, the number k is the dimension of the signal process defined in the preceding section, and the number i is the number of the components of the signal process whose initial values are to be loaded into the MLPWOF at time $n = 0$. Note that the intial values to be loaded may also affect our selection of the signal components for recursive neural filtering. This is further discussed in our discussion on dynamical range reducers. The number m of the external inputs is set equal to the dimension of the measurement vector. We note that the main purpose of teacher-influenced feedbacks is to load an initial signal state into the MLPWOF.

The number j of free outputs, the number L of layers, and the number of nodes in each layer are to be selected parsimoniously to achieve both a satisfactory filtering accuracy and a good generalization ability of the recursive neural filter. Three factors can be used in guiding the selection. First, if $L \geq 2$ and if j and the number n_1 of hidden nodes are sufficiently large, then the MLPWOF can approximate the minimum-variance filter to any accuracy. This is proven in Section V, and suggests that we start with $L = 2$. Second, free feedbacks are used to carry the conditional statistics required for "measurement update" of the signal estimate that are not teacher-influenced feedbacks in the MLPWOF, "measurement update" being a terminology borrowed from the Kalman theory. For instance, if the conditional error covariance of the signal estimate is expected to be sufficient and is not teacher-influenced-feedbacked, the number of free feedbacks is $k(k+1)/2$. Third, hidden nodes provide the MLPWOF's capability to approximate the dynamics of the minimum-variance filter, a combination of the "measurement update" and "time update" equations.

IV. Target Outputs for Training

For simplicity of discussion, we will assume in the rest of this paper that neither the signal process to be estimated nor the teacher-influenced feedbacks contain any conditional error statistics and that all the teacher-influenced outputs are feedbacked after a unit-time delay.

For training an MLPWIN into a recursive neural filter, a training data set is required that consists of realizations of the signal/information processes, which we denote by $\{(x_n(s), y_n(s)), n = 1, \ldots, N, s \in S\}$, where N is a time length believed long enough for the realizations to represent all possible dynamics of the signal and information processes. It is assumed that the set S is a random sample and adquately reflects the joint probability distributions of the signal and information processes x_n and y_n.

In general, the target outputs used in constructing an error criterion for training a neural network are what the neural network outputs are intended to approximate. If an MLPWOF is used as a minimum-variance recursive neural filter, the target output at time n should then be the minimum-variance estimate of the signal at the same time, which is known to be the conditional expectation $E[x_n \,|\, y^n]$ of x_n given the past realization $y^n = \{y_n \,|\, 1 \leq i \leq n\}$ of the information process. However, such minimum-variance estimates are difficult to obtain in most nonlinear filtering situations. Fortunately, the signals themselves can be used instead of their minimum-variance estimates. This is a consequence of the following probability properties.

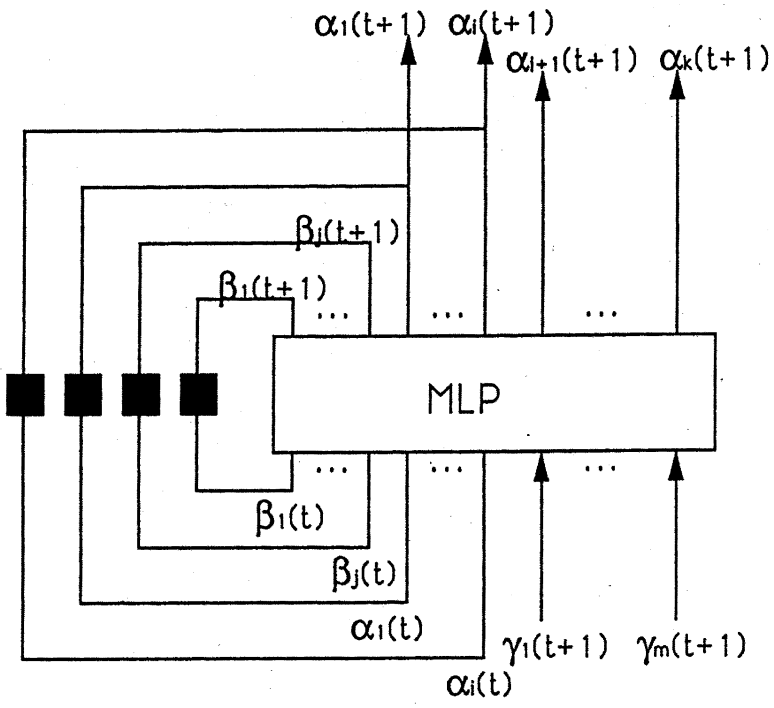

Fig. 2. An MLPWOF.

Recall that the collection of all k-dimensional random vectors $x = [x_1, \ldots, x_k]^T$ is a Hilbert space L_2 with the norm $E[\|x\|^2] = E[\sum_{l=1}^{k} x_l^2]$, where E denotes the expectation or the average over the sample space Ω with the σ-field $\mathcal{A}$ of events and the probability measure P.

By simple calculation, we have for any Borel-measurable function f of y^n

$$\begin{aligned} C' &:= \frac{1}{N}\sum_{n=1}^{N} E[\|x_n - f(y^n)\|^2] \\ &= \frac{1}{N}\sum_{n=1}^{N}\{E[\|x_n - E[x_n \,|\, y^n]\|^2] \\ &\quad + E[\|E[x_n \,|\, y^n] - f(y^n)\|^2]\}. \end{aligned} \tag{4}$$

Let the output of an MLPWOF with weights w (including biases) be denoted by $\alpha(t, w, s)$ after the MLPWOF has input the measurement realization $y^n(s) = \{y_i(s) \,|\, 1 \le i \le n\}$ one at a time in the given order. Notice that $\alpha(n, s)$ is a Borel function of y^n. Consider the error function constructed with the training data set S

$$C(w) = \frac{1}{N|S|}\sum_{s\in S}\sum_{n=1}^{N} \|x_n(s) - \alpha_n(s)\|^2. \tag{5}$$

Under the assumption that the training data set S is a random sample and adquately reflects the joint probability distributions of the signal and information processes x_n and y_n, we see from (4) that

$$\begin{aligned} C(w) &= \frac{1}{N|S|}\sum_{s\in S}\sum_{n=1}^{N} \|x_n(s) - E[x_n \,|\, y^n(s)]\|^2 \\ &\quad + \frac{1}{N|S|}\sum_{s\in S}\sum_{n=1}^{N} \|E[x_n \,|\, y^n(s)] - \alpha_n(s)\|^2. \end{aligned}$$

Since the first term on the right does not involve w, this shows that minimizing $C(w)$ by the variation of w is equivalent to minimizing $1/(N|S|)\sum_{s\in S}\sum_{n=1}^{N} \|\alpha_n(s) - E[x_n \,|\, y^n(s)]\|^2$ by the same. In other words, the error function (5) can be used as the error criterion for training an MLPWOF into a minimum-variance recursive neural filter.

V. Minimum-Variance Recursive Neural Filters

Recall the MLPWOF architecture described in Section III. We observe that adding a node to a hidden layer decreases $\min_w C(w)$ and so does adding a feedback, free or teacher-influenced. We are now ready to state and prove the main theorem of this paper.

Theorem: Let an N_x-dimensional stochastic process x_n and an N_y-dimensional stochastic process $y_n, n = 1, \ldots, N$, be defined on a probability space $(\Omega, \mathcal{A}, P)$. Assume that the range $\Lambda := \{y_n(s) \,|\, n = 1, \ldots, N, s \in \Omega\} \subset R^m$ is compact and the second moments $E[\|x_n\|^2], n = 1, \ldots, N$, are finite.

1) Consider an MLPWOF with a single hidden layer with M nodes, $N_y(N-1)$ free output feedbacks, no teacher-influenced output feedback, N_y external input terminals for receiving y_n, and N_x teacher-influenced output terminals for outputing an estimate of x_n at time n. Let the N_x-dimensional teacher-influenced output vector at time n be denoted by $\alpha_n(M)$ after having received $y_n, i = 1, \ldots, n$ one by one in the given order at the N_y external input terminals, and let the weights of the MLPWOF be denoted by w. Then the sequence

$$r(M) := \min_w \frac{1}{N}\sum_{n=1}^{N} E[\|\alpha_n(M) - E[x_n \,|\, y^n]\|^2] \tag{6}$$

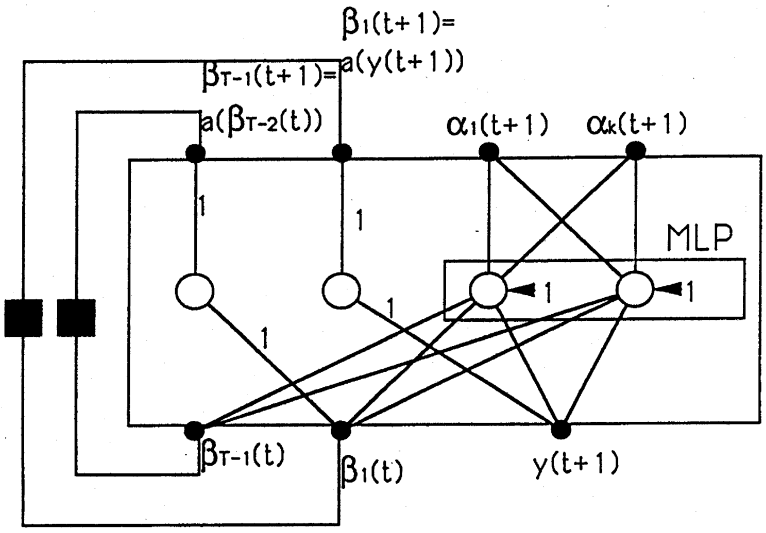

Fig. 3. An MLPWOF for proving existence of a recursive neural filter.

is monotone decreasing to zero as the number M of hidden nodes of the MLPWOF approaches infinity.

2) If any number of outputs from the teacher-influenced output terminals in the above MLPWOF are feedbacked to an equal number of teacher-influenced feedback input terminals after a unit-time delay, then the sequence (6) is also monotone decreasing to zero as the number M of hidden nodes of the MLPWOF approaches infinity.

Proof: Note that the MLPWOF with M hidden nodes is a special MLPWOF, that with $M+1$ hidden nodes and with the weights leading into and out of the extra hidden node set equal to zero. The minimization in (6) for the general MLPWOF with $M+1$ hidden nodes by the variation of all its weights including those leading into and out of the extra hidden node, thus, has $r(M+1)$ smaller than or equal to $r(M)$. Therefore, the sequence $r(M), M = 1, 2, \ldots,$ is monotone decreasing. Since $\|\alpha_n(M) - E[x_n \,|\, y^n]\|^2 \geq 0$, it follows that all $r(M)$ are bounded from below by zero. Hence, the sequence $r(M)$ converges and we denote the limit by $r(\infty)$.

To prove that $r(\infty) = 0$, it suffices to show that for any $\epsilon > 0$, there is an integer M' such that $|r(M')| < \epsilon$. This will be shown in the following for the case in which the information process y_n is scalar-valued, i.e., $m = 1$. The proof for $m > 1$ is similar and omitted. □

As illustrated in Fig. 3, $N-1$ free output feedbacks can be so interconnected that for $n = 0, \ldots, N-1, \beta_{1(n+1)} = a(y_{n+1})$ and for $i = 2, \ldots, N-1$

$$\beta_{i(n+1)} = a\left(\beta_{(i-1)n}\right). \tag{7}$$

These nodes are initialized at $\beta_{i0} = 0, i = 1, \ldots, N-1$. At time $n+1$, the lagged feedbacks from them are

$$(\beta_{N-1n}, \ldots, \beta_{n+1n}, \beta_{nn}, \beta_{n-1n}, \ldots, \beta_{1n}) = \left(0, \ldots, 0, a^{\circ n}(y_1), a^{\circ(n-1)}(y_2), \ldots, a(y_n)\right)$$

where $a^{\circ n}$ denotes the n-fold composition $a \circ a \circ \cdots \circ a$ of a. Including both these feedbacks and the input node, we have the N-dimensional vector at time $n+1$

$$z_{n+1} = \left[\beta_{(N-1)n}, \ldots, \beta_{1n}, y_{n+1}\right]^T$$

to use as the inputs to an MLP with one hidden layer.

Recall that $E[x_{n+1} \,|\, a^{\circ n}(y_1), a^{\circ(n-1)}(y_2), \ldots, a(y_n), y_{n+1}]$ is a Borel measurable function of $a^{\circ n}(y_1), \ldots, a(y_n)$ and y_{n+1}. Let us denote it by $\lambda_{n+1}(a^{\circ n}(y_1), \ldots, a(y_n), y_{n+1})$. Now let us consider the mapping $f : R^N \to R^k$ defined by the equation at the bottom of the page. Let us consider also the

$$f(\xi_1, \ldots, \xi_N) = \begin{cases} \lambda_1(\xi_N), & \text{if } \xi_1 = \cdots = \xi_{N-1} = 0, \quad \xi_N \neq 0 \\ & \text{and} \quad \lambda_1(\xi_N) \text{ defined;} \\ \lambda_2(\xi_{N-1}, \xi_N), & \text{if } \xi_1 = \cdots = \xi_{N-2} = 0, \xi_{N-1} \neq 0, \\ & \xi_N \neq 0 \quad \text{and} \quad \lambda_2(\xi_{N-1}, \xi_N) \text{ defined;} \\ \vdots & \\ \lambda_N(\xi_1, \ldots, \xi_N), & \text{if } \xi_1 \neq 0, \ldots, \xi_N \neq 0 \quad \text{and} \\ & \lambda_N(\xi_1, \ldots, \xi_N) \text{ defined;} \\ 0, & \text{otherwise.} \end{cases}$$

measures μ_n defined on the Borel sets in R^N by (denoting $d\xi_1 \times \cdots \times d\xi_N$ by $d\boldsymbol{\xi}$)

$$\mu_1(d\xi) = \begin{cases} \frac{1}{N}P(y_1 \in d\xi_N), & \text{if } 0 \in d\xi_n, \text{for } n < N, \\ 0, & \text{otherwise;} \end{cases}$$

$$\mu_2(d\xi) = \begin{cases} \frac{1}{N}P(a(y_1) \in d\xi_{N-1}, y_2 \in d\xi_N), \\ \quad \text{if } 0 \in d\xi_n, \text{ for } n < N-1, \\ 0, & \text{otherwise;} \end{cases}$$

$$\vdots$$

$$\mu_N(d\xi) = \frac{1}{N}P\left(a^{\circ(N-1)}(y_1) \in d\xi_1, \ldots, a(y_{N-1}) \in d\xi_{N-1}), y_N \in d\xi_T\right).$$

Notice that $\mu = \mu_1 + \cdots + \mu_N$ is also a measure on the Borel sets in R^N and that

$$\begin{aligned} &\int \|f(\xi_1, \ldots, \xi_N)\|^2 \mu(dbf\xi) \\ &\quad = \frac{1}{N}\left\{\int \|\lambda_1(\xi_N)\|^2 \mu_1(d\boldsymbol{\xi}) \right. \\ &\qquad + \int \|\lambda_2(\xi_{N-1}, \xi_N)\|^2 \mu_2(d\boldsymbol{\xi}) \\ &\qquad + \cdots \\ &\qquad \left. + \int \|\lambda_N(\xi_1, \ldots, \xi_N)\|^2 \mu_N(d\boldsymbol{\xi})\right\} \\ &\quad = \frac{1}{N}\left\{E[\|E[x_1 \,|\, y_1]\|^2] + \cdots \right. \\ &\qquad \left. + E\left[\left\|E\left[x_N \left| a^{\circ(N-1)}(y_1), \ldots, y_N\right.\right]\right\|^2\right]\right\} \\ &\quad \le \frac{1}{N}\{E[\|x_1\|^2] + \cdots + E[\|x_N\|^2]\} \\ &\quad < \infty. \end{aligned}$$

Since $y_n, n = 1, \ldots, N$ are assumed to have compact ranges, it is easy to see that $a^{\circ n}(y_i), n$ and $i = 1, \ldots, N$ all have compact ranges. Hence, there is a compact set W in R^N such that $\mu(W) = 1$.

Consider now the normed space $L_2(R^N, \mu)$ with the norm defined by $[\int \|g(\xi_1, \ldots, \xi_N)\|^2 \mu(d\boldsymbol{\xi})]^{1/2}$ for each k-dimensional function $g \in L_2(R^N, \mu)$. By Corollary 2.2 of the paper [37] by Hornik *et al.*, the k-dimensional functions represented by all the MLPs with k output nodes and a single layer of hidden nodes are dense in $L_2(R^N, \mu)$. It then follows that for any $\epsilon > 0$, there is such an MLP that the function $g(\xi_1, \ldots, \xi_N)$ that it represents satisfies

$$\int \|f(\xi_1, \ldots, \xi_N) - g(\xi_1, \ldots, \xi_N)\|^2 \mu(d\boldsymbol{\xi}) < \epsilon.$$

Let us translate the left side of this inequality from $L_2(R^N, \mu)$ back to the probability space $(\Omega, \mathcal{A}, P)$

$$\begin{aligned} &\int \|f(\xi_1, \ldots, \xi_N) - g(\xi_1, \ldots, \xi_N)\|^2 \mu(d\boldsymbol{\xi}) \\ &\quad = \frac{1}{N}\left\{\int \|\lambda_1(\xi_N) - g(0, \ldots, 0, \xi_N)\|^2 \mu_1(d\boldsymbol{\xi}) \right. \\ &\qquad + \int \|\lambda_2(\xi_{N-1}, \xi_N) - g(0, \ldots, 0, \xi_{N-1}, \xi_N)\|^2 \\ &\qquad \times \mu_2(d\boldsymbol{\xi}) \\ &\qquad + \cdots \\ &\qquad \left. + \int \|\lambda_N(\xi_1, \ldots, \xi_N) - g(\xi_1, \ldots, \xi_N)\|^2 \mu_N(d\boldsymbol{\xi})\right\} \\ &\quad = \frac{1}{N}\sum_{n=1}^{N} E\left[\left\|E[x_n \,|\, y^n] \right.\right. \\ &\qquad \left.\left. - g(0, \ldots, 0, a^{\circ(n-1)} \cdot (y_1), a^{\circ(n-2)}(y_2), \ldots, y_n)\right\|^2\right] \end{aligned}$$

where the last equality holds, because the σ-fields generated by y^n and $\{a^{\circ(n-1)}(y_1), a^{\circ(n-2)}(y_2), \ldots, y_n\}$ are identical due to the monotonicity of a.

We notice that the foregoing approximation MLP, $g(\xi_1, \ldots, \xi_N)$, and the $N-1$ free output feedbacks (7) discussed earlier form an MLPWOF with a single hidden layer. The number M of hidden nodes of this MLPWOF is the sum of that of the MLP and $N-1$. Recalling (6), we have

$$\begin{aligned} r(M) \le \frac{1}{N}\sum_{n=1}^{N} E\left[\left\|E[x_n \,|\, y^n] - g(0, \ldots, 0, \right.\right. \\ \left.\left. a^{\circ(n-1)}(y_1), a^{\circ(n-2)}(y_2), \ldots, y_n)\right\|^2\right] < \epsilon \end{aligned}$$

since $r(M)$ minimizes the error function in (6) by the variation of the weights of an MLPWOF with exactly the same architecture as for the MLPWOF constructed earlier. This completes the proof of 1).

The MLPWOF with teacher-influenced feedbacks that is described in 2) can be viewed as a general case of the MLPWOF described in 1): if the weights on those teacher-influenced feedback connections of the former are set equal to zero, it is reduced to an MLPWOF without the teacher-influenced feedbacks. Therefore, the sequence $r(M)$ of the MLPWOF with teacher-influenced feedbacks is also monotone decreasing to zero as the number M of hidden nodes of the MLPWOF approaches infinity. This completes the proof of 2).

As discussed earlier on, free output feedbacks are used to carry conditional statistics required for optimal filtering. Therefore, the number of free output feedbacks needed depends very much on the properties and the probability distributions of the signal and information processes x_n and y_n individually and jointly. For instance, if x_n and y_n satisfy 1) and 2) with f and h being linear in x_n, G constant, and x_0, v_n, and w_n Gaussian, the free output feedbacks needed for minimum variance filtering are the conditional mean and covariances of x_n given y^n, regardless of N.

VI. Dynamical Range Reducers

A dynamical range reducer is a preprocessor of an RNN that dynamically transforms at least one component of the exogenous input process such as an information process and sends the resulting process to at least one input terminal of the RNN. A possible benefit of using a dynamical range reducer is a reduction of the valid input range or approximation capability required of the RNN so as to ease the RNN size and training data requirements and thereby lessen the

training difficulty. Another possible benefit is an enhancement of the generalization capability of the RNN beyond the length of time for which the training data are available.

A basic scheme for dynamically transforming the ith component y_{in} of an exogenous input process y_n is to subtract some estimate $\hat{y}_{\text{in}}$ of y_{in} from y_{in} at every time point n. A scheme that generates a causal estimate $\hat{y}_{\text{in}}$ is called an auxiliary estimator of y_{in}. The resulting difference, $y_{\text{in}} - \hat{y}_{\text{in}}$, is used at time n as the ith component of the input vector to the RNN. A device that comprises an auxilliary estimator to generate an auxiliary estimate $\hat{y}_{\text{in}}$, and a subtractor to perform the subtraction, $y_{\text{in}} - \hat{y}_{\text{in}}$, is called a dynamical range reducer by estimate subtraction.

The purpose of the auxiliary estimate is only to reduce the input range of the RNN. Therefore, the auxiliary estimate does not have to be very accurate. However, notice that the difference process $y_{\text{in}} - \hat{y}_{\text{in}}, i = 1, \ldots$ is causally equivalent to the information process $y_{\text{in}}, i = 1, \ldots$ only if y_{i1} is used jointly with the difference process. If a dynamical range reducer by estimate subtraction is employed for recursive neural filtering, there are three cases to be considered.

Case 1) The initial signal x_1 is a fixed vector. In this case, the RNN will learn to integrate this vector into the estimates produced by the RNN during its training.

Case 2) The signal process is observable from the difference process $y_{\text{in}} - \hat{y}_{\text{in}}, i = 1, \ldots$. In this case, the estimate produced by the RNN will still converge to that of the optimal estimate given the original information process. It only takes a little longer.

Case 3) A good estimate of the initial signal is available. In this case, we use an MLPWOF into which the initial signal can be loaded through the teacher-influenced output feedback terminals of the MLPWOF. If the estimate of the initial signal is good enough to determine a good estimate of the first measurement y_1, the difference process resulting from the dynamical range reducer together with the initial signal is about causally equivalent to the information process, and the filtering performance of the MLPWOF is about the same as that of the optimal filter based on the original information process y.

Three types of dynamical range reducer by estimate subtraction are given as examples in the following:

A. Range Reducers by Differencing

If an information process y_n consists of the vector values, at discrete time points, of a continuous continuous-time process, then the vector value y_{n-1} is a "reasonably good" estimate of the vector value y_n. This observation motivated a simple, yet effective way to reduce the range of the measurements, when two consecutive measurements y_{n-1} and y_n are not too far apart.

Consider the recursive neural filter depicted in Fig. 4. A differencer that consists of a unit time delay and a subtractor is concatenated at a input terminal of an RNN. At each time point n, the differencer subtracts the preceding measurement

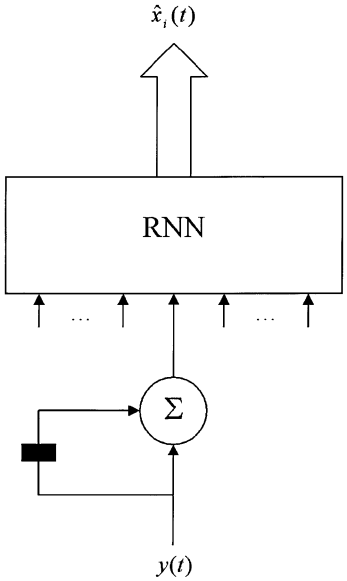

Fig. 4. A range reducer by differencing.

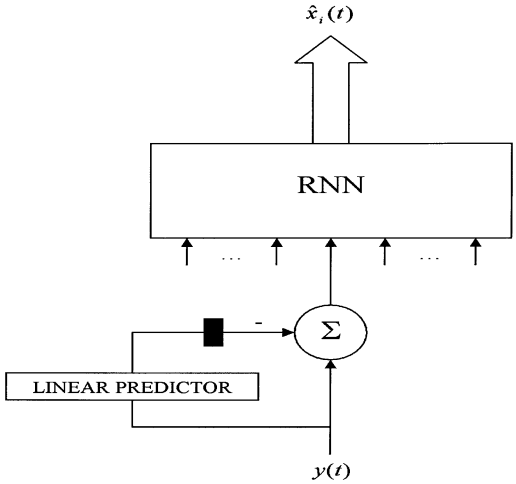

Fig. 5. A range reducer by linear prediction.

$y_{i(n-1)}$ from the current measurement y_{in} and feeds the difference $y_{\text{in}} - y_{i(n-1)}$ to the ith input terminal of the RNN.

There are three ways to initialize the differencer. One way is to start the recursive neural filter at $n = 2$, the ith component of the first input vector to the RNN being $y_{i2} - y_{i1}$ and the first output vector of the RNN being $\hat{x}_2$. The second way is to determine an initialization value for y_{i0} jointly with the weights w and initial dynamical state v of the RNN in training. In the operation of the recursive neural filter, the ith component of the first input vector to the RNN is $y_{i1} - y_{i0}$. The third way is to use the best available estimate $\bar{y}_{i0}$ of y_{i0} and then use $y_{i1} - \bar{y}_{i0}$ as the ith component of the first input vector to the RNN consistently in the training and operation of the recursive neural filter.

B. Range Reducers by Linear Prediction

Consider the recursive neural filter depicted in Fig. 5, where one dynamical range reducer is shown, which consists of a linear predictor, a unit time delay device, and a subtractor.

The linear predictor inputs the ith component of the input process to the recursive neural filter, which input process is the information process y_n, and outputs a prediction $\hat{y}_{i(n+1)}$ of $y_{i(n+1)}$. After a unit time delay, the preceding prediction $\hat{y}_{\text{in}}$ is now subtracted from y_{in} by the subtractor. The resulting difference $y_{\text{in}} - \hat{y}_{\text{in}}$ is then input to the RNN at its ith input terminal.

A range reducer by differencing is obviously a special range reducer by linear prediction, in which the estimate $\hat{y}_{i(n+1)}$ generated by the linear predictor is simply y_{in}. A general linear predictor is written as $\hat{y}_{\text{in}} = \sum_{j=1}^{J} c_j y_{i(t-j)}$, where J is a fixed positive integer called the order of the linear predictor, and c_j are the linear predictor coefficients [38]. Realizations of the ith component y_{in} of the information process, which are part of the training data to be discussed in the sequel, are used to determine $c_j, j = 1, 2, \ldots, J$ so that the linear predictor $\hat{y}_{\text{in}} = \sum_{j=1}^{J} c_j y_{i(t-j)}$ predicts y_{in} in the standard least-squares sense. A fast and stable algorithm for this can be found in [39]. Some other algorithms can be found in [38].

There are two ways to initialize the linear predictor. One way is to start the recursive neural filter at $n = J + 1$, the ith component of the first input vector to the RNN being $y_{i(J+1)} - \hat{y}_{i(J+1)}$ and the first output vector of the RNN being $\hat{x}_{J+1}$. The second way is to determine J initialization values, $y_{i(-J+1)}, y_{i(-J+2)}, \ldots, y_{i0}$ jointly with the weights w and initial dynamical state v of the RNN in training. In the operation of the recursive neural filter, the i th component of the first input vector to the RNN is $y_{\text{in}} - \sum_{j=1}^{J} c_j y_{i(t-j)}$.

C. Range Reducers by Model-Aided Prediction

Assume that a model in the form of (1) and (2) for the signal process x_n and information process y_n is available and that a dynamical range reducer is required for the ith component of the information process, which is the input process to the recursive neural filter.

Consider the recursive neural filter depicted in Fig. 6, where only one dynamical range reducer is shown. The unit time delay device, function f, function h_i, and subtractor constitute a dynamical range reducer. The output $\hat{x}_{n-1}$ of the recursive neural filter at time $n-1$ is the best estimate of the signal x_{n-1} that is available to the recursive neural filter. Given the model (1) and (2), it is obvious that a good prediction $\hat{\hat{x}}_n$ of x_n is $\hat{\hat{x}}_n = f(\hat{x}_{n-1}, n-1)$ and a good prediction $\hat{y}_{\text{in}}$ of y_{in} is $\hat{y}_{\text{in}} = h_i(f(\hat{x}_{n-1}, n-1), n)$. At time $n = 1$, the estimate $\hat{x}_0$ is set equal to an *a priori* estimate $\bar{x}_0$ of x_0 and the prediction $\hat{y}_{i1}$ of y_{i1} is then equal to $h_i(f(\bar{x}_0, 0), 1)$. The *a priori* estimate $\bar{x}_0$ is simply the best estimate of x_0 that is available. Its accuracy as an estimate of x_0 is not critical. However, once it is chosen, it should be consistently used in both the training and the operation of the recursive neural filter. An alternative way to determine $\hat{y}_{i1}$ is to determine it jointly with the weights w and initial dynamical state v of the RNN in minimizing a selected training criterion to be discussed in the sequel.

We notice that this dynamical range reducer provides the RNN with extra feedbacks of its outputs into its inputs, in addition to the feedbacks inside the RNN. This should be taken into consideration in training the RNN together with the dynamical range reducer.

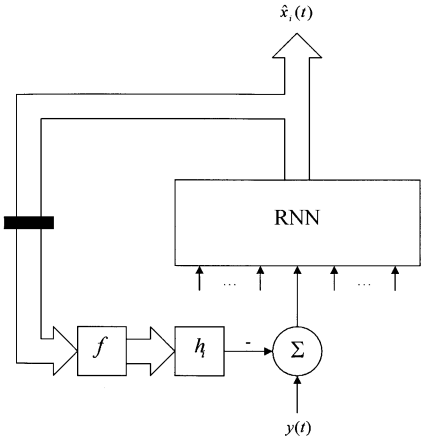

Fig. 6. A range reducer by model-aided prediction.

VII. Dynamical Range Extenders

The function of a dynamical range extender is to extend and/or transform the output range of an RNN to cover the range of a signal process. Given a signal process, a purpose of using a dynamical range extender is to reduce the valid output range and/or approximation capability required of an RNN so as to ease the RNN size and training data requirements and thereby lessen the training difficulty. In the following, we will simplify our discussion by restricting it to a dynamical range extender's function of extending the output range of an RNN to achieve the purpose of reducing the valid output range required of the RNN. It will be appreciated that another function of a dynamical range extender is simply to transform an RNN's output range, without necessarily extending it, to achieve another purpose of reducing the approximation capability required of the RNN, without necessarily reducing the valid output range required of the same RNN.

A basic scheme for dynamically transforming the output range of an output node, say, node i in layer L, of an RNN is to add some estimate $\hat{\hat{x}}_{\text{in}}$ of the desired output x_{in} for the same output node to the node's actual output β_{in}^L at every time point n. The resulting sum $\beta_{\text{in}}^L + \hat{\hat{x}}_{\text{in}}$ is used as the ith component $\hat{x}_{\text{in}}$ of the output vector $\hat{x}_n$ of the neural filter at time n. Thus, the "actual desired output" for the output node is $x_{\text{in}} - \hat{\hat{x}}_{\text{in}}$ at time n, whose range is expected to be smaller than the range of x_{in} , provided that the estimate $\hat{\hat{x}}_{\text{in}}$ is "good." The estimate $\hat{\hat{x}}_{\text{in}}$ will be called an auxiliary estimate of x_{in} and a scheme that generates this estimate $\hat{\hat{x}}_{\text{in}}$ will be called an auxiliary estimator. A device that comprises such an auxiliary estimator and an adder will be called dynamical range extender by estimate addition, which is a dynamical transformer of the output process β_{in}^L.

For an RNN with a range extender by estimate addition to approximate the optimal filter to any accuracy, a fundamental requirement for the dynamical range extender is that the estimate $\hat{\hat{x}}_{\text{in}}$ be a function of the measurements $y_i, i = 1, 2, \ldots, n$ for $n = 1, 2, \ldots, N$. The proof of this statement is easy, and is outlined as follows. Given ε , let $\hat{\hat{x}}_{\text{in}}$ and the optimal estimate of x_{in} be approximated, respectively, to within

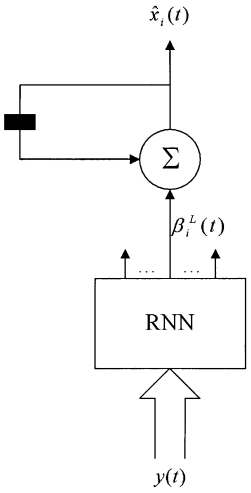

Fig. 7. A range extender by accumulation.

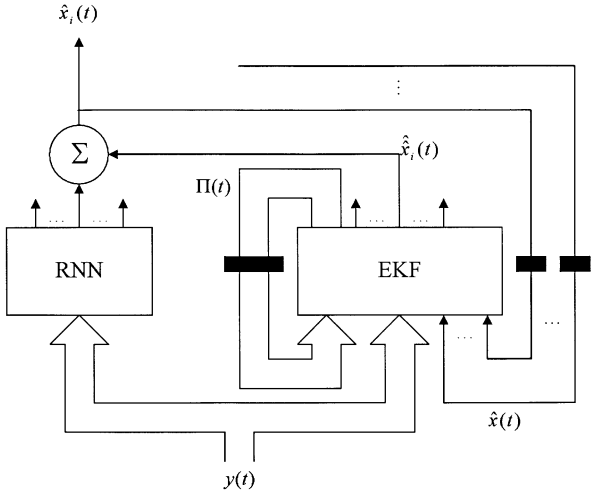

Fig. 8. A range extender by Kalman filtering.

$\varepsilon/2$ by two RNNs with respect to the minimum-variance error criterion. Then their difference approximates $x_{\text{in}} - \hat{\hat{x}}_{\text{in}}$ to within ε. Since the combination of the two RNNs can be viewed as an RNN, the proof is completed.

Five types of range extender by estimate addition, whose auxiliary estimators have different levels of estimation accuracy and different levels of computational cost, are given in the following.

A. Range Extenders by Accumulation

If a signal process x_n consists of the vector values, at discrete time points, of a slowly varying continuous continuous-time process, then the vector value x_{n-1} is a good approximate of x_n, and a good estimate of x_{n-1} is a "reasonably good" estimate of the vector value x_n. This observation motivated a simple, yet effective way to extend the output range of an RNN in a neural filter, when two consecutive signals x_{n-1} and x_n are not too far apart.

Consider the neural filter depicted in Fig. 7. Only one accumulator used as a dynamical range extender is shown. The accumulator, consisting of a unit time delay device and an adder, is concatenated directly to output node i of the RNN. At each time point n, the accumulator adds the output β_{in}^L of the RNN to the accumulator's output $\hat{x}_{i(n-1)}$ at the preceding time point $n-1$. Thus the accumulator accumulates all the outputs of output node i of the RNN from $n=1$ onward plus the initial accumulation denoted by $\hat{x}_{i0}$. Mathematically, the accumulator is described simply by

$$\hat{x}_{\text{in}} = \beta_{\text{in}}^L + \hat{x}_{i(n-1)}, \quad n = 1, 2, \ldots, N. \tag{8}$$

Here, the RNN actually estimates the difference $x_{\text{in}} - \hat{x}_{i(n-1)}$, which is expected to have a much smaller range than does x_{in}, if the two consecutive signals x_{n-1} and x_n are not too far apart. If a good *a priori* estimate $\bar{x}_{i0}$ is given of x_{i0}, it should be used as the initial accumulation $\hat{x}_{i0}$. Otherwise, the initial accumulation $\hat{x}_{i0}$ can be determined together with the weights and/or parameters w and the initial dynamical state v of the RNN in minimizing a training criterion for the neural filter.

Note that $\hat{x}_{i(n-1)}$ is a function of $y_1, y_2, \ldots, y_{n-1}$, for $n = 1, 2, \ldots, N$. Viewing $\hat{x}_{i(n-1)}$ as an estimate $\hat{\hat{x}}_{\text{in}}$ of x_{in} that is added to the output β_{in}^L of output node i, we see that the accumulator is a range extender by estimate addition, which satisfies the fundamental requirement for a range extender by estimate addition as stated earlier on. An accumulator used as a dynamical range extender will be called a range extender by accumulation.

B. Range Extenders by Kalman Filtering

Assume that a model in the form of (1) and (2) for the signal process x_n and measurement process y_n is available and that a dynamical range extender is required for every output node of the RNN in a neural filter.

Consider the neural filter depicted in Fig. 8, where only one dynamical range extender for an output node is shown. Recall a solid square represents a unit time delay device. The EKF and the adder constitute a dynamical range extender. The EKF uses the output $\hat{x}_{n-1}$ from the neural filter, the error covariance matrix Π_{n-1} from itself, and the measurement vector y_n to produce an estimate $\hat{\hat{x}}_n$ of x_n. The RNN is then employed only to estimate the difference $x_{\text{in}} - \hat{\hat{x}}_{\text{in}}$ for the ith signal component. The range of the difference is expected to be much smaller than x_{in}, if $\hat{\hat{x}}_{\text{in}}$ is a good estimate of x_{in}. Denoting the ith output of the RNN by β_{in}^L, the estimate $\hat{x}_{\text{in}}$ of x_{in} generated by the neural filter is $\hat{x}_{\text{in}} = \hat{\hat{x}}_{\text{in}} + \beta_{\text{in}}^L$.

The EKF equations are

$$\Pi_{n\,|\,n-1} = F_{n-1}\Pi_{n-1}F_{n-1}^T + G_{n-1}(\hat{x}_{n-1}) \cdot Q_{n-1}G_{n-1}^T(\hat{x}_{n-1}) \tag{9}$$

$$\hat{\hat{x}}_{n\,|\,n-1} = f_{n-1}(\hat{x}_{n-1}) \tag{10}$$

$$\Omega_n = H_n^T\Pi_{n\,|\,n-1}H_n + R_n \tag{11}$$

$$L_n = \Pi_{n\,|\,n-1}H_n\Omega_n^{-1} \tag{12}$$

$$\hat{\hat{x}}_n = \hat{\hat{x}}_{n\,|\,n-1} + L_n[y_n - h(\hat{\hat{x}}_{n\,|\,n-1})] \tag{13}$$

$$\Pi_n = \Pi_{n\,|\,n-1} - \Pi_{n\,|\,n-1}H_n\Omega_n^{-1}H_n^T\Pi_{n\,|\,n-1} \tag{14}$$

where $F_{n-1} = (\partial f_{n-1}(x)/\partial x)\,|_{x=\hat{x}_{n-1}}$, and $H^T = (\partial h_n(x)/\partial x)\,|_{x=\hat{\hat{x}}_{n\,|\,n-1}}$. At $n=0$, we initialize the neural

filter by setting $\hat{\hat{x}}_0 = \bar{x}_0$ and $\Pi_0 = \Pi_0$ to activate the above EKF equations.

Note that $\hat{\hat{x}}_n$ is indeed a function of $y_1, y_2, \ldots, y_n$, for $n = 1, 2, \ldots, N$, satisfying the aforementioned fundamental requirement for a range extender by estimate addition.

We stress here that the above EKF equations involve the outputs of the RNN through $\hat{x}_{n-1}$ and are, thus, not the standard EKF equations. If an RNN is properly selected and trained, the estimate $\hat{\hat{x}}_n$ generated by the above EKF equations is better than the standard extended Kalman estimate.

If the signal process x_n is only a part of the process described by (1), the estimate $\hat{x}_{n-1}$ generated by the neural filter is then only a part of the "preceding estimate" used in the EKF equations for (1) and (2). The rest of the "preceding estimate" used is necessarily the corresponding components of the preceding EKF estimate generated by these EKF equations.

The RNN with range extenders by Kalman filtering can be viewed as using the RNN to make up the nonlinear part of an optimal estimate that is ignored by the EKF due to linearization.

C. Range Extenders by Feedforward Kalman Filtering

The only difference between this type of dynamical range extender and the preceding type, namely, range extenders by Kalman filtering, is that the "preceding estimate" $\hat{x}_{n-1}$ used in the EKF equations (9)–(14) is now replaced by $\hat{\hat{x}}_{n-1}$. Thus, the EKF equations used here are the standard EKF equations without the involvement of $\hat{x}_{n-1}$ generated by an RNN. This type of dynamical range extender requires that the EKF does not diverge and is useful for improving existing EKFs.

A range extender by feedforward Kalman filtering is usually inferior to a range extender by Kalman filtering. However, including a range extender by feedforward Kalman filtering in a neural system does not incur much extra computation for training the recursive neural filter. Since the EKF equations used here do not involve the RNN in the neural system, the only special treatment in training for a range extender by feedforward Kalman filtering is to use $x_{\text{in}} - \hat{\hat{x}}_{\text{in}}, n = 1, 2, \ldots, N$ instead of $x_{\text{in}}, n = 1, 2, \ldots, N$ as the target output process for the output node i involved in the RNN.

D. Range Extenders by Linear Prediction

Consider the recursive neural filter depicted in Fig. 9 where only one dynamical range extender is shown. The one shown is a range extender by estimate addition that consists of a linear predictor, a unit time delay device, and an adder, and is concatenated to output node i of an RNN. The estimate $\hat{\hat{x}}_{\text{in}}$, to be added to β_{in}^L to yield $\hat{x}_{\text{in}}$, i.e., $\hat{x}_{\text{in}} = \beta_{\text{in}}^L + \hat{\hat{x}}_{\text{in}}$, is generated by a linear predictor. A range extender by accumulation can be viewed as a special case in which $\hat{x}_{i(n-1)}$ is used as the predicted value of $\hat{x}_{\text{in}}$.

A better estimate of $\hat{x}_{\text{in}}$ than $\hat{x}_{i(n-1)}$, which is used in a range extender by accumulation, can be obtained by the linear predictor $\hat{\hat{x}}_{\text{in}} = \sum_{j=1}^{J} c_j \hat{x}_{i(n-j)}$, where J is a fixed positive integer called the order of the linear predictor and c_j are the linear predictor coefficients [38]. However, before

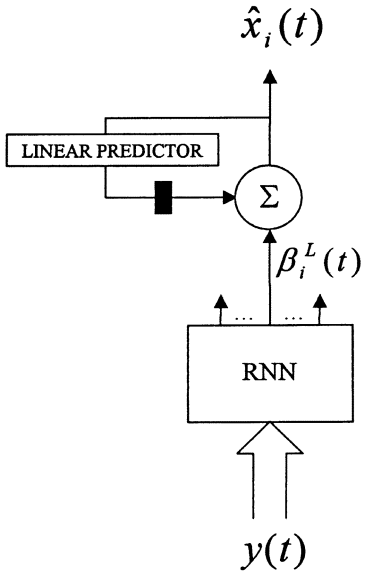

Fig. 9. A range extender by linear prediction.

both the RNN and the linear predictor are fully specified, the sequence $\hat{x}_{\text{in}} = \beta_{\text{in}}^L + \hat{\hat{x}}_{\text{in}}, n = 1, 2, \ldots, N$ is not available, preventing us from applying standard methods to determine the linear predictor coefficients $c_j, j = 1, 2, \ldots, J$ and thereby specify the linear predictor. To get around this difficulty, we may determine the linear predictor coefficients for predicting the signal x_{in} instead. More specifically, we use realizations of the ith component x_{in} of the signal process, which realizations are part of the training data to be discussed in the sequel, to determine $c_j, j = 1, 2, \ldots, J$, so that the linear finite-impulse response (FIR) filter $\sum_{j=1}^{J} c_j x_{i(n-j)}$ predicts x_{in} in the standard least-squares sense. A fast and stable algorithm for this can be found in [38].

Then we use these coefficients $c_j, j = 1, 2, \ldots, J$ as the coefficients in the linear predictor $\hat{\hat{x}}_{\text{in}} = \sum_{j=1}^{J} c_j \hat{x}_{i(n-j)}$ for predicting $\hat{x}_{\text{in}}$. The resulting linear predictor is expected to generate good estimate $\hat{\hat{x}}_{\text{in}}$ of $\hat{x}_{\text{in}}$, provided $\hat{x}_{\text{in}}$ mimics x_{in} closely.

To initialize the linear predictor at $n = 1$, we need the initialization values $\hat{x}_{i(-J+1)}, \hat{x}_{i(-J+2)}, \ldots, \hat{x}_{i0}$ in both the training and the operation of the recursive neural filter. If the signals $x_{i(-J+1)}, x_{i(-J+2)}, \ldots, x_{i0}$ are available at $n = 1$ in the operation of the recursive neural filter in the application under consideration, we may include realizations of $x_{i(-J+1)}, x_{i(-J+2)}, \ldots, x_{i0}$ in the training data set in addition to those of $x_{i1}, x_{i2}, \ldots, x_{iN}$. In training, a realization of $x_{i(-J+1)}, x_{i(-J+2)}, \ldots, x_{i0}$ is used as the initialization values $\hat{x}_{i(-J+1)}, \hat{x}_{i(-J+2)}, \ldots, \hat{x}_{i0}$.

If the signals $x_{i(-J+1)}, x_{i(-J+2)}, \ldots, x_{i0}$ are not available at time $n = 1$ in the operation of the recursive neural filter in the application under consideration, we may use a starter-filter to process $y_{-J+1}, y_{-J+2}, \ldots, y_0$ for estimating $x_{-J+1}, x_{-J+2}, \ldots, x_0$. The resulting estimates are then used as the initialization values. Here, of course, we need the measurements $y_{-J+1}, y_{-J+2}, \ldots, y_0$, which are either extra measurements or the measurements $y_1, y_2, \ldots, y_J$ with the time scale shifted. We may employ an EKF as a starter filter. Since J is usually small and the ranges of the signal and

information processes cannot be very large over the time interval $n = -J+1, -J+2, \ldots, 0$, a simple recursive neural filter without dynamical range extenders and reducers is expected to work nicely here as a starter filter.

Now holding the coefficients c_j constant, we synthesize the training data into an RNN. If the resulting recursive neural filter, including the linear predictor, adder, and RNN, works satisfactorily, the process of designing a recursive neural filter is completed. Otherwise, we may increase J and repeat the above process of determining c_j and then synthesizing an RNN again or we may adjust the values of $c_j, j = 1, 2, \ldots, J$ together with the weights w and the initial dynamical state v of the RNN by minimizing the training criterion further, using the existing values of c_j, w, and v as the initial guess in the minimization process.

E. Range Extenders by Feedforward Linear Estimation

In many filtering environments, a linear FIR filter can be used to process the information process to obtain a rather good estimate of the signal process. Using such a linear filter as an auxiliary estimator, a range extender by estimate addition is obtained, which will be called a range extender by feedforward linear estimation. Such a dynamical range extender is shown in Fig. 10. The input vector to its auxiliary estimator at time n is the m-dimensional measurement vector y_n and the output vector of the auxiliary estimator at the same time is the auxiliary estimate of those components of the signal process. Let the vector with those components be denoted by z_n and the auxiliary estimate of z_n by denoted by $\hat{\hat{z}}_n$.

The auxiliary estimator is a linear estimator described by $\hat{\hat{z}}_n = \sum_{j=0}^{J-1} C_j y_{n-j}$, where J denotes the order of the linear estimator and C_j, for $j = 0, 1, \ldots, J-1$, are the coefficient matrices. Using the components of $z_n(s)$ corresponding to those of $\hat{\hat{z}}_n$ as the desired output, and $y_n(s)$ as the input for each $s \in S$, the coefficient matrices are determined by minimizing $\sum_{s \in S} \sum_{n=1}^{N} \| z_n(s) - \sum_{j=0}^{J-1} C_j y_{t-j}(s) \|^2$, where $\| \cdot \|$ is the Euclidean norm. Assuming that $y(i, s)$ is zero for $i \leq 0$ and $s \in S$, the recursive least-squares algorithm in [38] can be applied to calculate the coefficient matrices $C_j, j = 0, 1, \ldots, J-1$.

The estimate $\hat{z}_{\text{in}}$ of z_{in} generated by the recursive neural filter depicted in Fig. 10 is the sum of $\hat{\hat{z}}_{\text{in}}$ and β_{in}^L, the ith output of the RNN. To initialize the linear estimator in operation at $n = 1$, we need the initialization values for $y_{-J+1}, y_{-J+2}, \ldots, y_0$. If they are not available, we may set them equal to zero in synthesizing the recursive neural filter and then in operating the recursive neural filter. An alternative way to determine the initialization values is to optimize them jointly with the weights of the recursive neural filter.

The auxiliary estimate $\hat{\hat{z}}_n$ generated by the linear estimator is fedforward to the adder. Since the linear estimator does not involve the RNN in its generation of $\hat{\hat{z}}_n$, the only special treatment in training for a range extender by feedforward linear estimation is to use $z_n - \hat{\hat{z}}_n, n = 1, 2, \ldots, N$ as the target output process for the RNN.

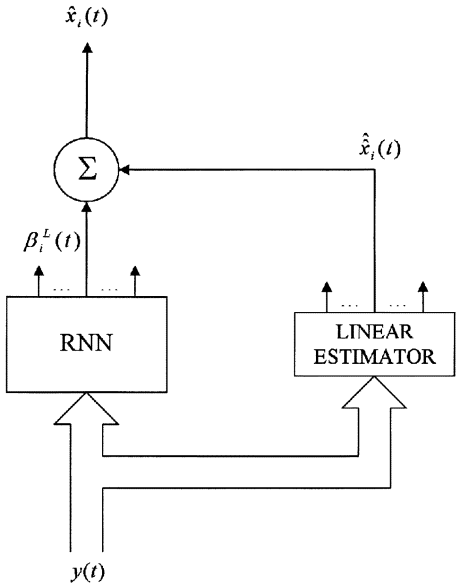

Fig. 10. A range extender by feedforward linear estimation.

VIII. Numerical Examples

Six simple numerical examples were worked out to demonstrate the numerical feasibility and effectiveness of the synthetic approach to optimal filtering using MLPWOFs and dynamical range transformers.

The first two show that recursive neural filters consisting of an MLPWOF with various combinations of free and teacher-influenced feedbacks outperform the EKF and the IEKF for nonlinear systems described by (1) and (2). It is also shown that if no estimate of initial signal is loaded into the recursive neural filter, free feedbacks are better than teacher-influenced feedbacks, and the recursive neural filters with only free feedbacks converge rather fast for the example signal/measurement systems.

The third example shows that the performance of a recursive neural filter consisting of an MLPWOF is indistinguishable from that of the KF for a linear system. In the fourth example, neither the signals nor the measurements can be described by (1) and (2). Nevertheless, the recursive neural filters work satisfactorily.

We note that the signal and measurement processes and the data for training recursive neural filters in the first four examples were deliberately chosen to be exactly the same as those used in [17] and [19] to synthesize and test recursive neural filters with an MLPWIN. An interested reader is encouraged to compare the performances of recursive neural filters based on an MLPWOF and an MLPWIN. Our observation is that under the same conditions, the difference is minimal.

In the fifth example, a range extender by accumulation is shown to outperform a static transformer, a sigmoidal function, for extending the output range of an MLPWIN, especially beyond the length of the period on which the training data were collected. The EKF and IEKF diverged in this example.

The sixth example compares the performances of an EKF and a recursive neural filter consisting of an MLPWIN, a range reducer by differencing, and a range extender by Kalman filtering. The difference in their performances is caused by the nonlinear part ignored by the EKF.

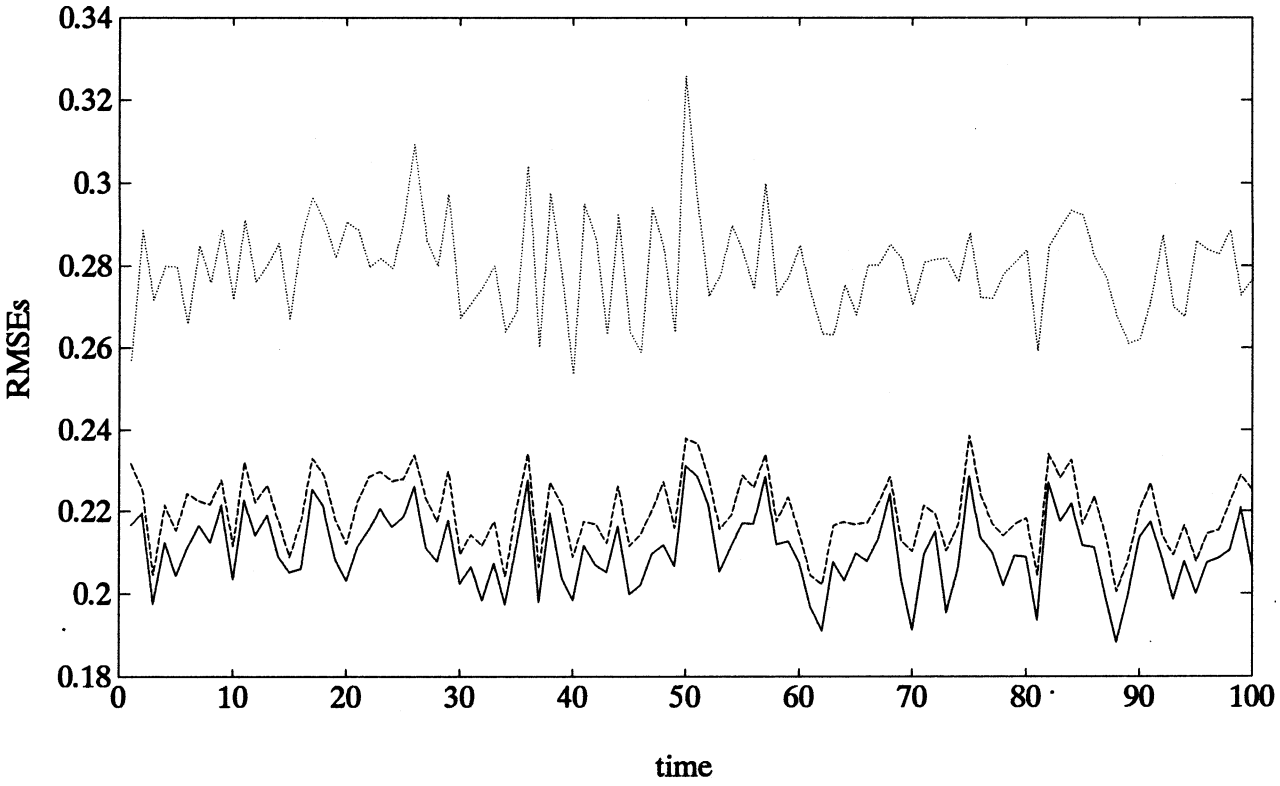

Fig. 11. RMSEs versus time of IEKF $(\cdots)$, NF$(7,1,0)(---)$, and NF$(7,0,1)(—)$ for Example 1. The RMSEs of them over 100 time points are 0.2800, 0.2200, and 0.2106, respectively.

In describing the signal and measurement processes in the examples, the symbols w_n and v_n denote statistically independent standard white Gaussian sequences with mean $E[w_n] = E[v_n] = 0$ and variance $E[w_n^2] = E[v_n^2] = 1$. The training data for each example are obtained by simulating the system and consist of batches of 200 realizations of 100 consecutive measurements and signals. A batch is used in a training session of 100 epochs (or sweeps). If the convergence of the weights is not satisfactory, a different batch is used in another training session. Continuing in this manner, we stop the training whenever the convergence is reached, i.e., the change in the mean square error $C(w)$ is less than $10^{-10}C(w)$.

After training, 500 Monte Carlo test runs were performed for each filter that is considered in each example. The root mean square error (RMSE) of each filter considered versus time is plotted for 100 time points in Examples 1 and 2, and for 120 time points to demonstrate the generalization ability of the recursive neural filter in Examples 1, 2, and 3. In Example 4, the true signal and the estimate produced by the recursive neural filter in a single run as well as the RMSE are plotted versus time for 120 time points.

In the following, we shall denote a recursive neural filter consisting of an MLPWOF with a single hidden layer of i nodes, j teacher-influenced feedbacks, and k free feedbacks by NF(i,j,k).

Example 1: The signal and measurement processes are

$$x_{n+1} = 1.1\exp\left(-2x_n^2\right) - 1 + 0.5w_n$$
$$y_n = x_n^3 + 0.1v_n$$

where x_0 is Gaussian with mean -0.5 and variance 0.1^2. Note that $x_{n+1} = 1.1\exp(-2x_n^2) - 1$ has a global attractor at $x_n = 0.0844$.

The EKF diverges. The RMSEs versus time for the IEKF and the recursive neural filters with one teacher-influenced NF$(7,1,0)$ or free feedback NF$(7,0,1)$ are plotted in Fig. 11. The total RMSEs of 500 Monte Carlo runs at each of 100 time points are 0.2800, 0.2200, and 0.2106, respectively. Recalling the common notion that the conditional mean and variance are necessary for propagating the optimal estimate even in the Kalman filtering for linear Gaussian signal and measurement processes, it is remarkable that an MLPWOF with a single feedback terminal can outperform an IEKF.

To compare the relative effects of teacher-influenced and free feedbacks, RMSEs of MLPWOFs with all combinations of three feedbacks are plotted in Fig. 12. The total RMSEs of NF$(7,3,0)$, NF$(7,2,1)$, NF$(7,1,2)$, and NF$(7,0,3)$ over 100 time points are 0.2369, 0.2124, 0.2121, and 0.2096, respectively. The results show that free feedbacks are usually better than teacher-influenced feedbacks.

That the performance of the recursive neural filter with only free feedbacks converges as the number of feedbacks increases is proven in Theorem V. It is numerically confirmed in Fig. 13. RMSEs of MLPWOFs with no teacher-influenced feedbacks and one, two and three free feedbacks are plotted. The total RMSEs of NF$(7,0,1)(\cdots)$, NF$(7,0,2)(---)$, and NF$(7,0,3)(—)$ over 120 time points are 0.2118, 0.2115 and 0.2113, respectively. Note that performances over the last 20 time points show the generalization ability of the recursive neural filters.

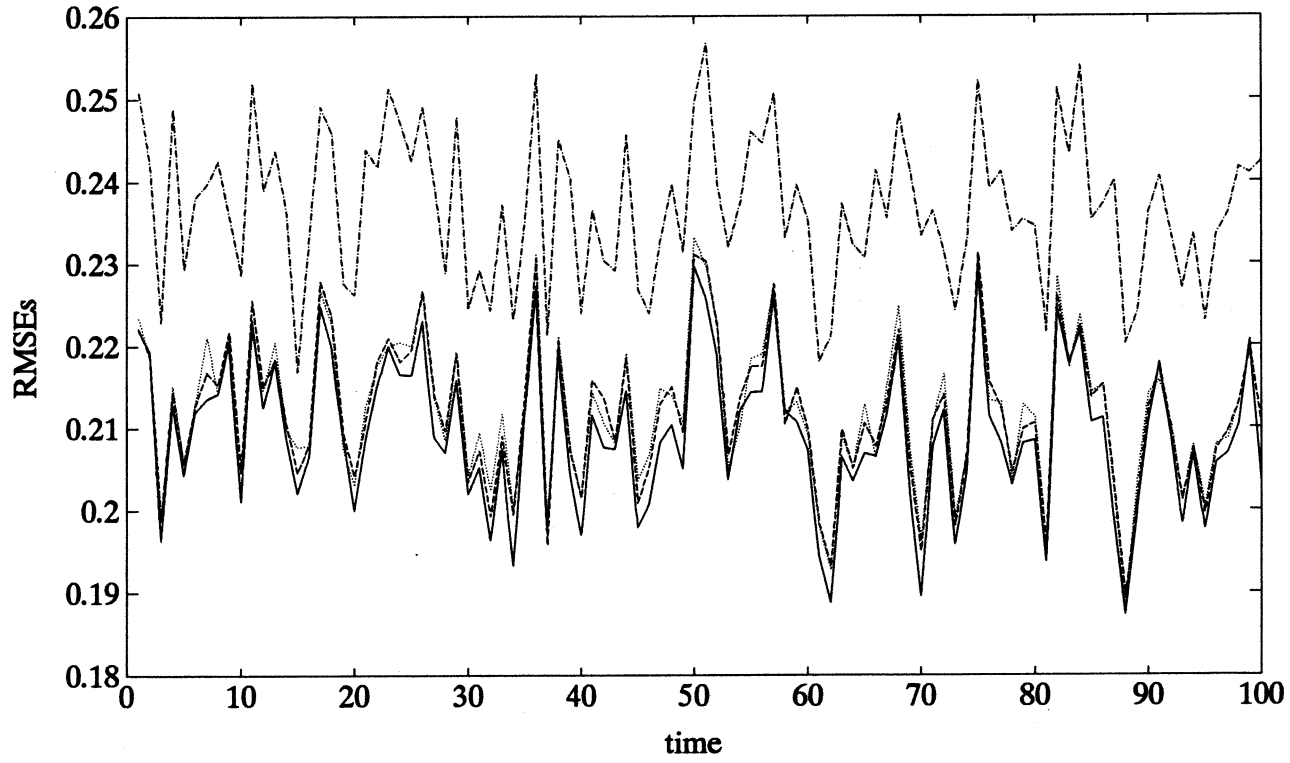

Fig. 12. RMSEs versus time of NF$(7,3,0)(--\cdot--)$, NF$(7,2,1)$ $(\cdots)$, NF$(7,1,2)$ $(---)$, and NF$(7,0,3)$ (—) for Example 1. The RMSEs of them over 100 time points are 0.2369, 0.2124, 0.2121, and 0.2096, respectively.

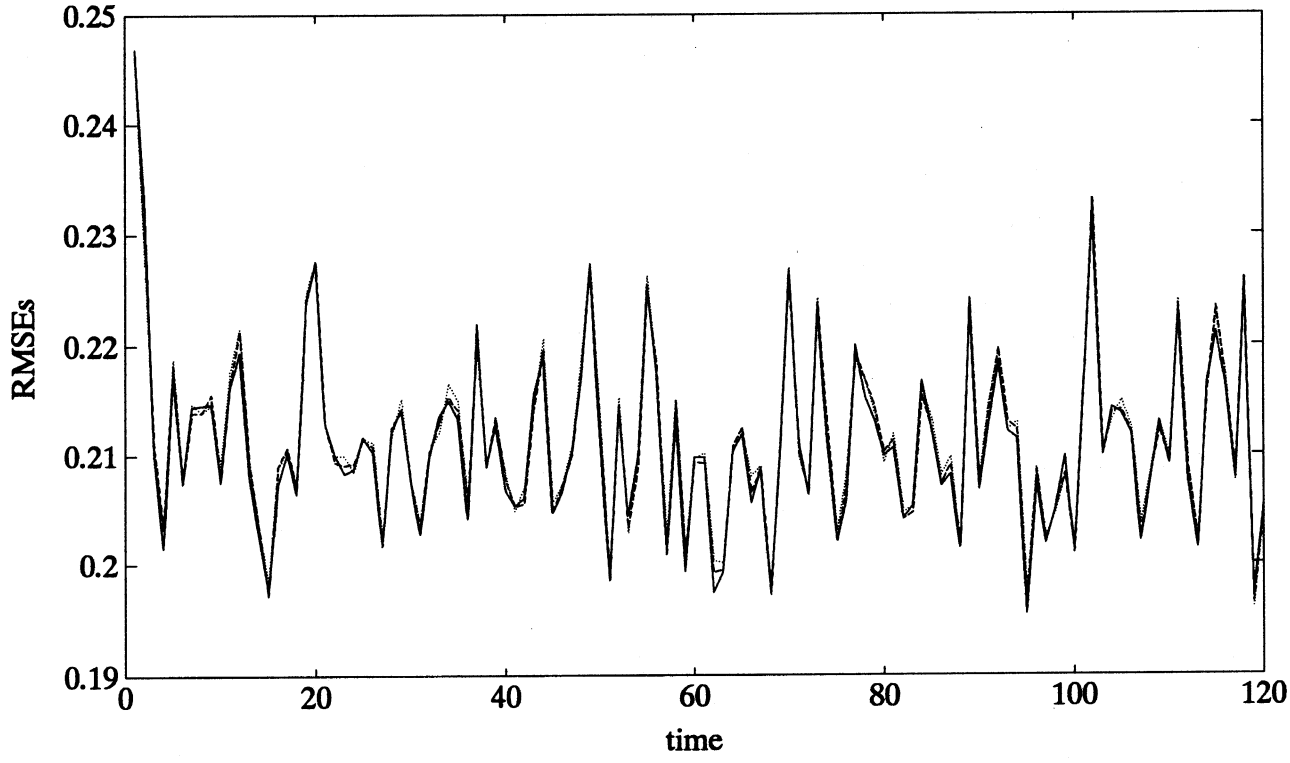

Fig. 13. RMSEs versus time of NF$(7,0,1)$ $(\cdots)$, NF$(7,0,2)$ $(---)$, and NF$(7,0,3)$ (—) for Example 1. The RMSEs of them over 120 time points are 0.2118, 0.2115, and 0.2113, respectively.

Example 2: The signal and measurement processes are

$$x_{n+1} = 1.7\exp\left(-2x_n^2\right) - 1 + 0.1w_n,$$
$$y_n = x_n^3 + 0.1v_n,$$

where x_0 is Gaussian with mean zero and variance 0.5^2. Note that $x_{n+1} = 1.7\exp(-2x_n^2) - 1$ is a chaotic process.

The RMSEs versus time for the EKF, IEKF, and recursive neural filters with one teacher-influenced NF$(7,1,0)$ or free feedback NF$(7,0,1)$ are plotted in Fig. 14. The total

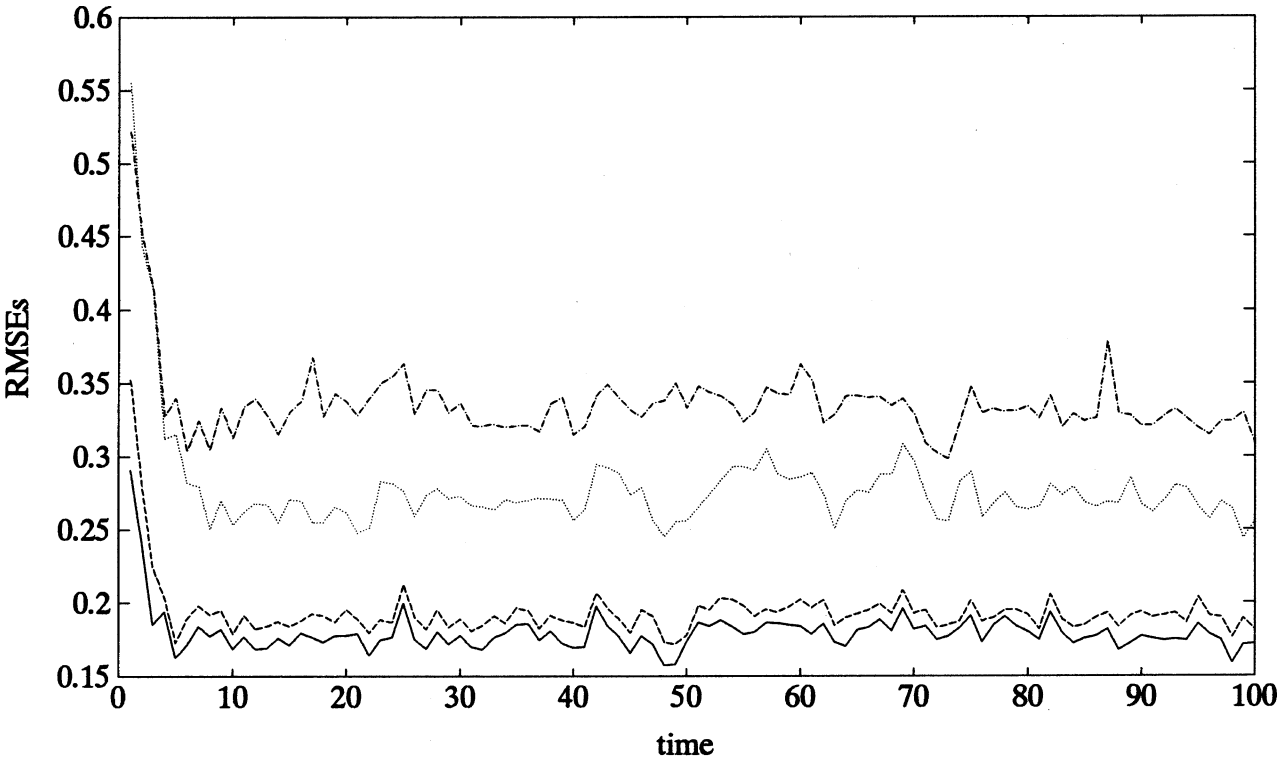

Fig. 14. RMSEs versus time of EKF $(- \cdot -)$, IEKF$(\cdots)$, NF$(7,1,0)$ $(---)$, and NF$(7,0,1)$ $(—)$ for Example 2. The RMSEs of them over 100 time points are 0.3371, 0.2807, 0.1942, and 0.1801, respectively.

RMSEs over 100 time points are 0.3371, 0.2807, 0.1942, and 0.1801, respectively. Recalling the common notion that the conditional mean and variance are necessary for propagating the optimal estimate even in the Kalman filtering for linear Gaussian signal and measurement processes, it is remarkable that an MLPWOF with a single feedback terminal can outperform an EKF and IEKF.

To compare the relative effects of teacher-influenced and free feedbacks, RMSEs of MLPWOFs with all combinations of three feedbacks are plotted in Fig. 15. The total RMSEs of NF$(7,3,0)$, NF$(7,2,1)$, NF$(7,1,2)$, and NF$(7,0,3)$ over 100 time points are 0.1836, 0.1833, 0.1774, and 0.1764, respectively. The results show that free feedbacks are better than teacher-influenced feedbacks.

That the performance of the recursive neural filter with only free feedbacks converges as the number of feedbacks increases is proven in Theorem V. It is numerically confirmed in Fig. 16. RMSEs of MLPWOFs with no teacher-influenced feedbacks and one, two, and three free feedbacks are plotted. The total RMSEs of NF$(7,0,1)$ $(\cdots)$, NF$(7,0,2)(- - -)$, and NF$(7,0,3)(—)$ over 120 time points are 0.1786, 0.1767, and 0.1763, respectively. Note that performances over the last 20 time points show the generalization ability of the recursive neural filters.

Example 3: The signal and measurement processes are

$$x_{n+1} = 0.9x_n + 0.2w_n$$
$$y_n = x_n + v_n$$

where x_0 is Gaussian with mean zero and variance 1^2.

The RMSEs versus time for the KF and NF$(0,2)$ are shown in Fig. 17. We note that the two lines are virtually the same.

Example 4: The signal and measurement processes are

$$x_{n+1} = 0.5x_n + 0.5\tanh(x_n + 0.5w_n)$$
$$y_n = x_n + 0.5x_n^3 v_n$$

where x_0 is Gaussian with mean zero and variance 0.5^2. Note that neither the signal nor the measurement process can be transformed into (1) or (2). The EKF and IEKF do not apply here.

The true signal and its estimates produced by the NF$(7,0,3)$ in a single run are shown in Fig. 18. The recursive neural filter tracks the true signal well except at a few time points, and is able to generalize beyond the time length of training data, 100.

The RMSEs versus time for the NF$(7,0,1)$, NF$(7,0,2)$, and NF$(7,0,3)$ are plotted in Fig. 19, which show that the performance of the recursive neural filter with only free feedbacks converges as the number of feedbacks increases. This confirms Theorem V.

Example 5: The signal and measurement processes are

$$x_{n+1} = x_n + 1.2\sin x_n + 1.21 + 0.2\xi_n \quad (15)$$
$$y_n = \sin x_n + 0.1\varepsilon_n \quad (16)$$

where ξ_n and ε_n are independent standard white Gaussian sequences, and $x_0 = 0$. The measurements y_n are confined

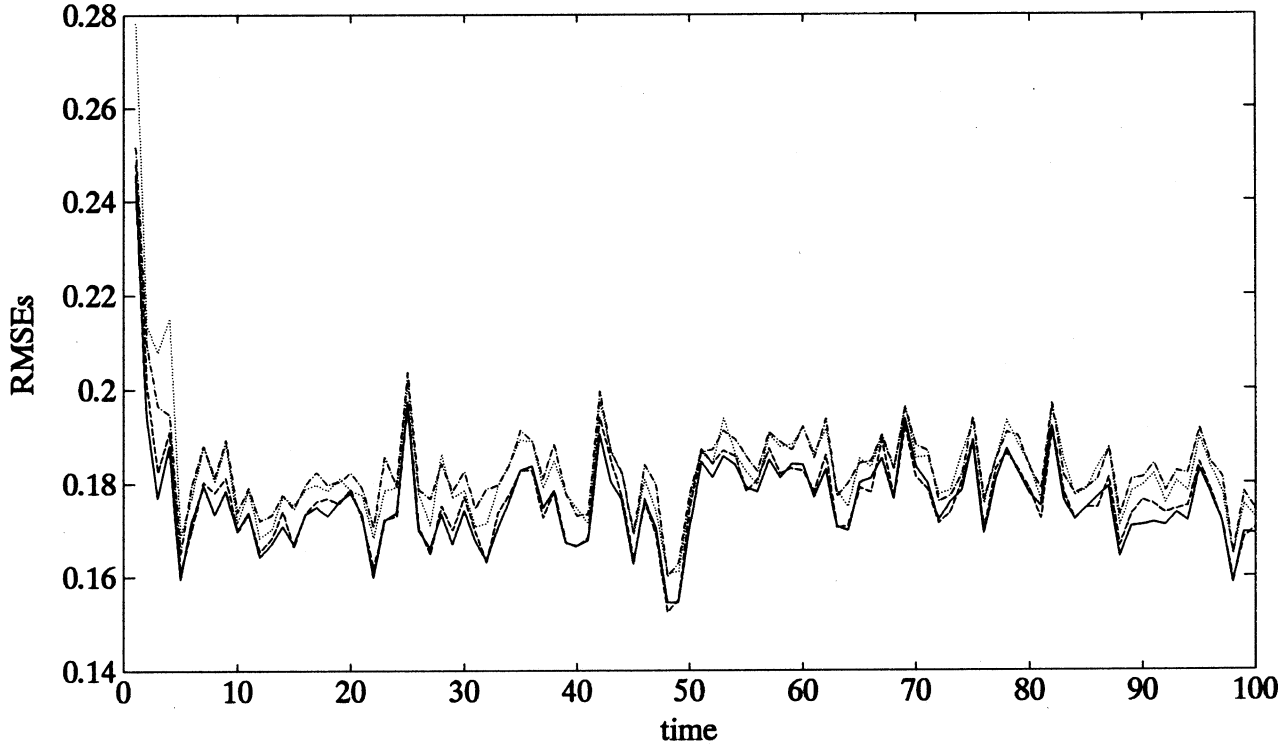

Fig. 15. RMSEs versus time of NF$(7,3,0)(- \cdot -)$, NF$(7,2,1)$ $(\cdots)$, NF$(7,1,2)$ $(---)$, and NF$(7,0,3)$(—) for Example 2. The RMSEs of them over 100 time points are 0.1836, 0.1833, 0.1774, and 0.1764, respectively.

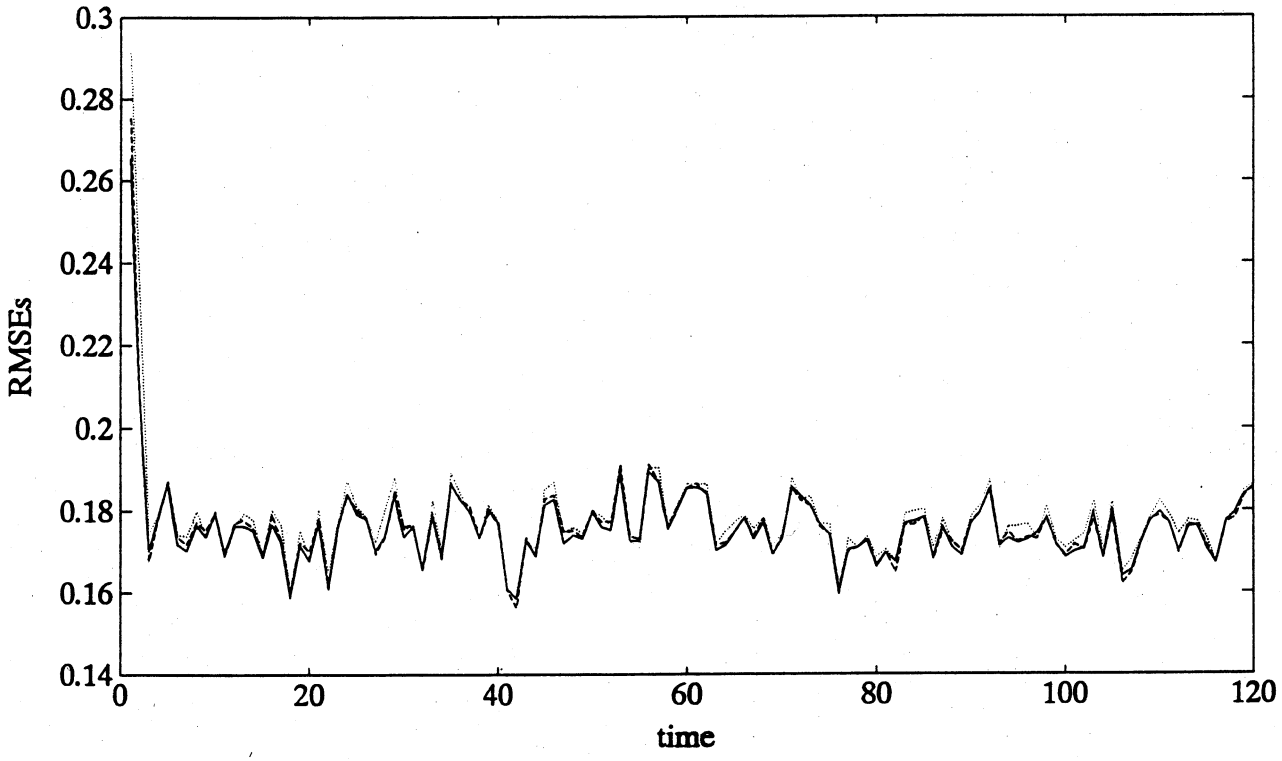

Fig. 16. RMSEs versus time of NF$(7,0,1)$ $(\cdots)$, NF$(7,0,2)$ $(---)$, and NF$(7,0,3)$ (—) for Example 2. The RMSEs of them over 120 time points are 0.1786, 0.1767, and 0.1763, respectively.

essentially in a compact domain. Nevertheless, the system remains locally observable because y_n is sensitive to the change of x_n no matter how large x_n is.

The EKF and the iterated EKFA failed badly here. Notice that the signal process keeps growing over time. A recursive neural filter without a dynamical range extender is expected

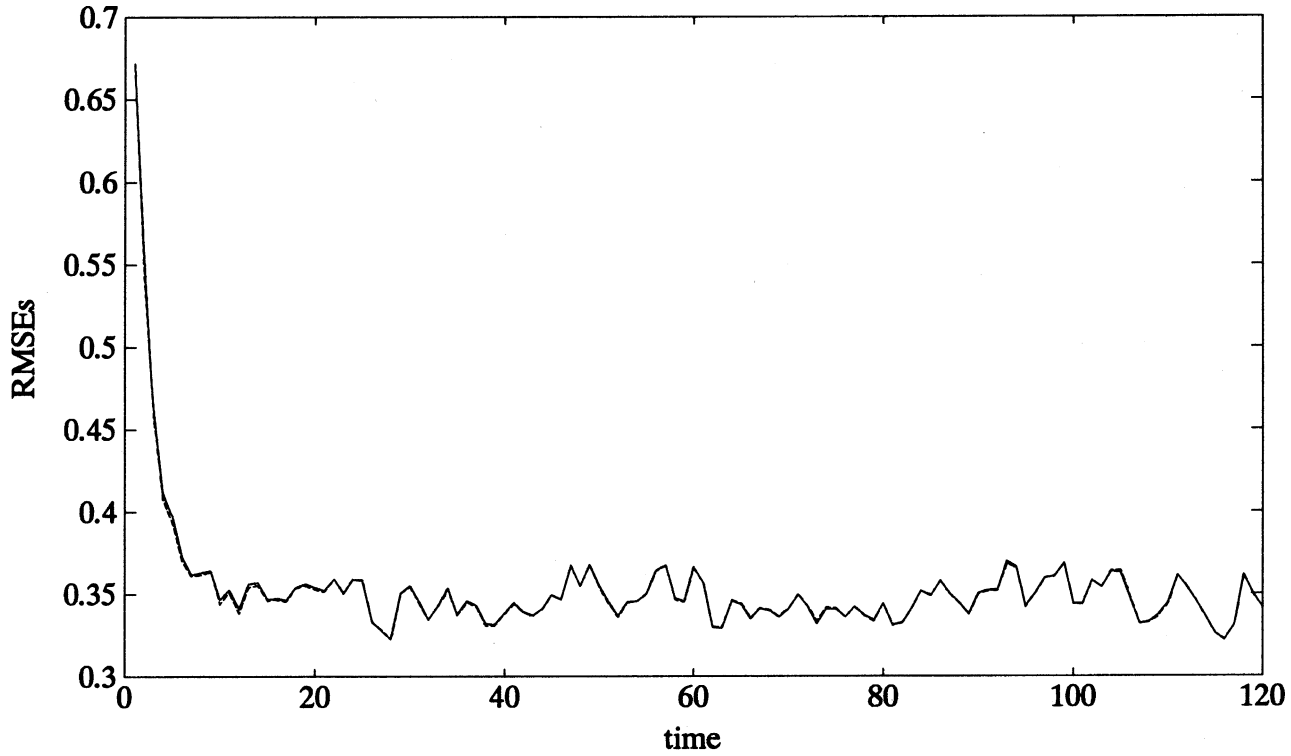

Fig. 17. RMSEs versus time of the KF $(---)$ and the NF$(7, 0, 2)$ $(—)$ for Example 3. The RMSEs of them over 120 time points are 0.3549 and 0.3555, respectively.

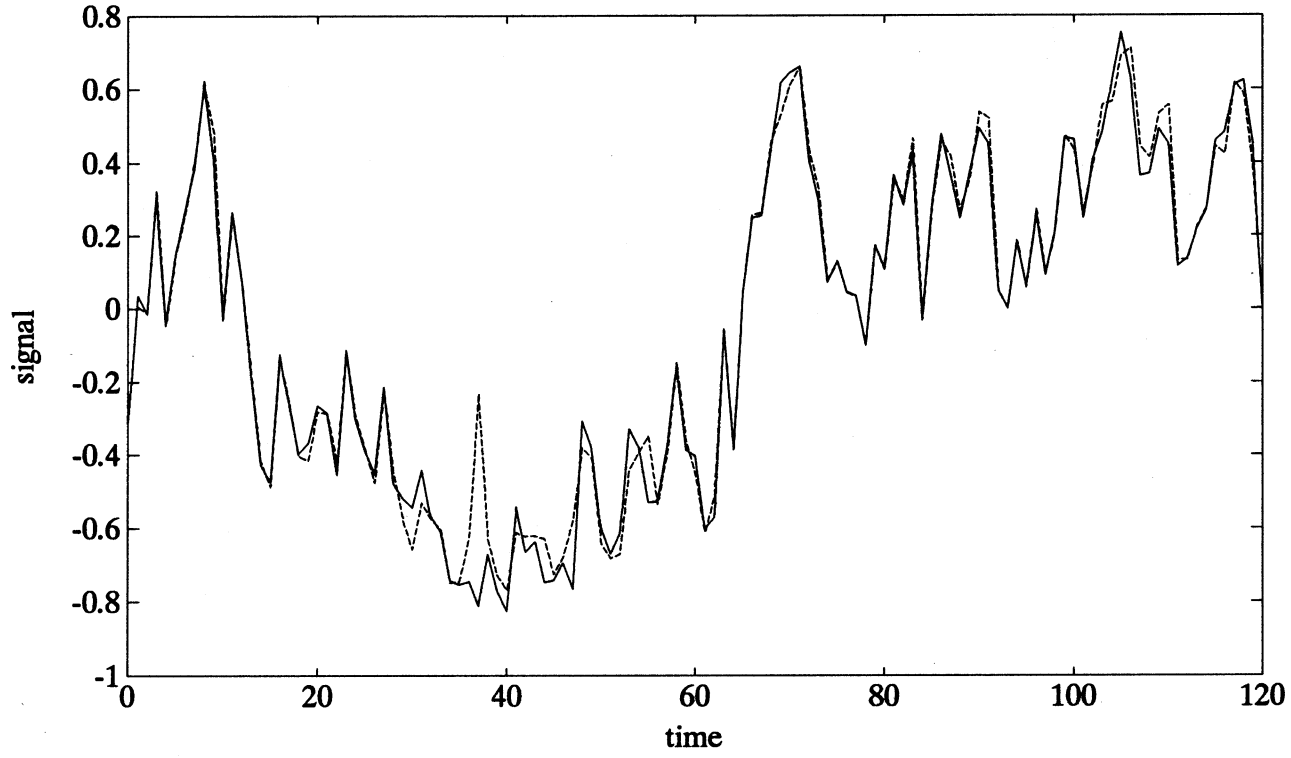

Fig. 18. The true signal $(—)$ and the estimate $(---)$ produced by the NF$(7, 0, 3)$ versus time for Example 4.

to fail beyond the time length of the training data. A range extender with accumulation is employed as a postprocessor of an MLPWIN with a single hidden layer of nine fully interconnected nodes for filtering y_n to estimate x_n. The filtering performances of this recursive neural filter, to be referred to as an NFWA, and a recursive neural filter consisting of an MLPWIN also with a single layer of nine nodes but with sigmoidal scaling to extend the RNN's output range, to be re-

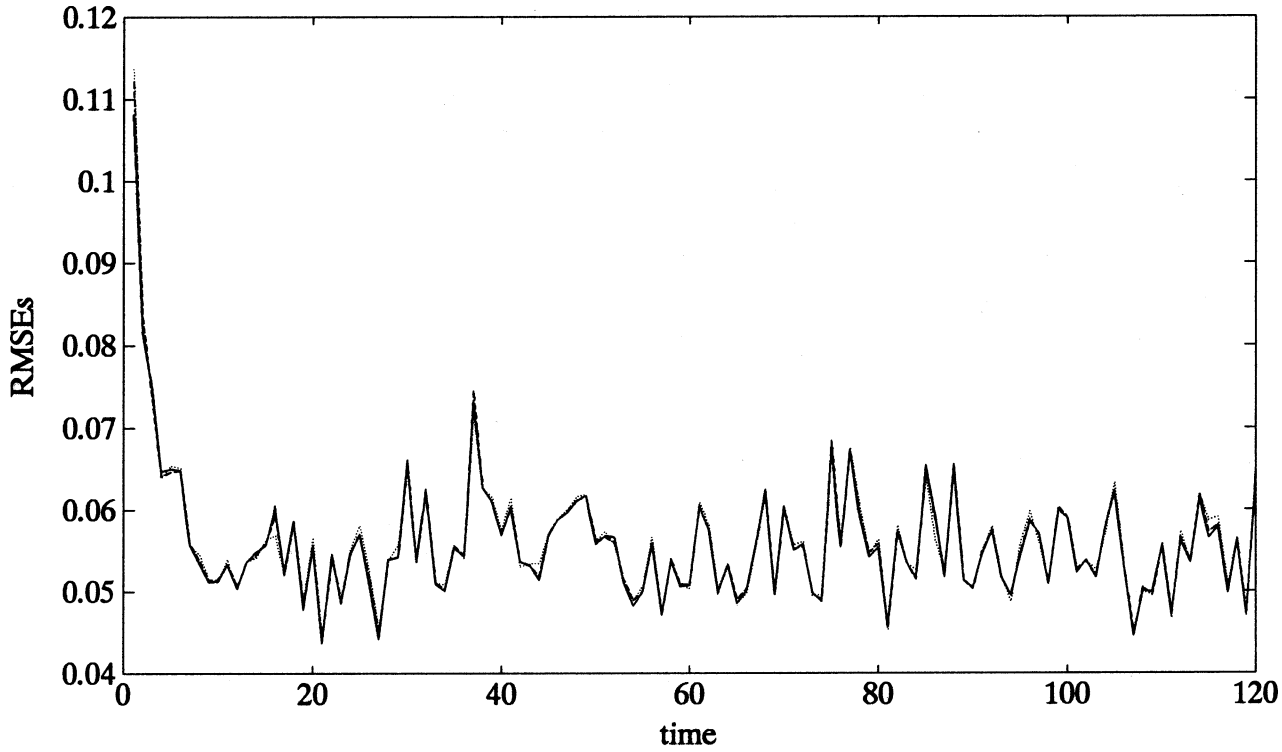

Fig. 19. RMSEs versus time of NF$(7,0,1)$ $(\cdots)$, NF$(7,0,2)$ $(---)$, and NF$(7,0,3)$ $(—)$ for Example 4. The RMSEs of them over 120 time points are 0.0568, 0.0567, and 0.0565, respectively.

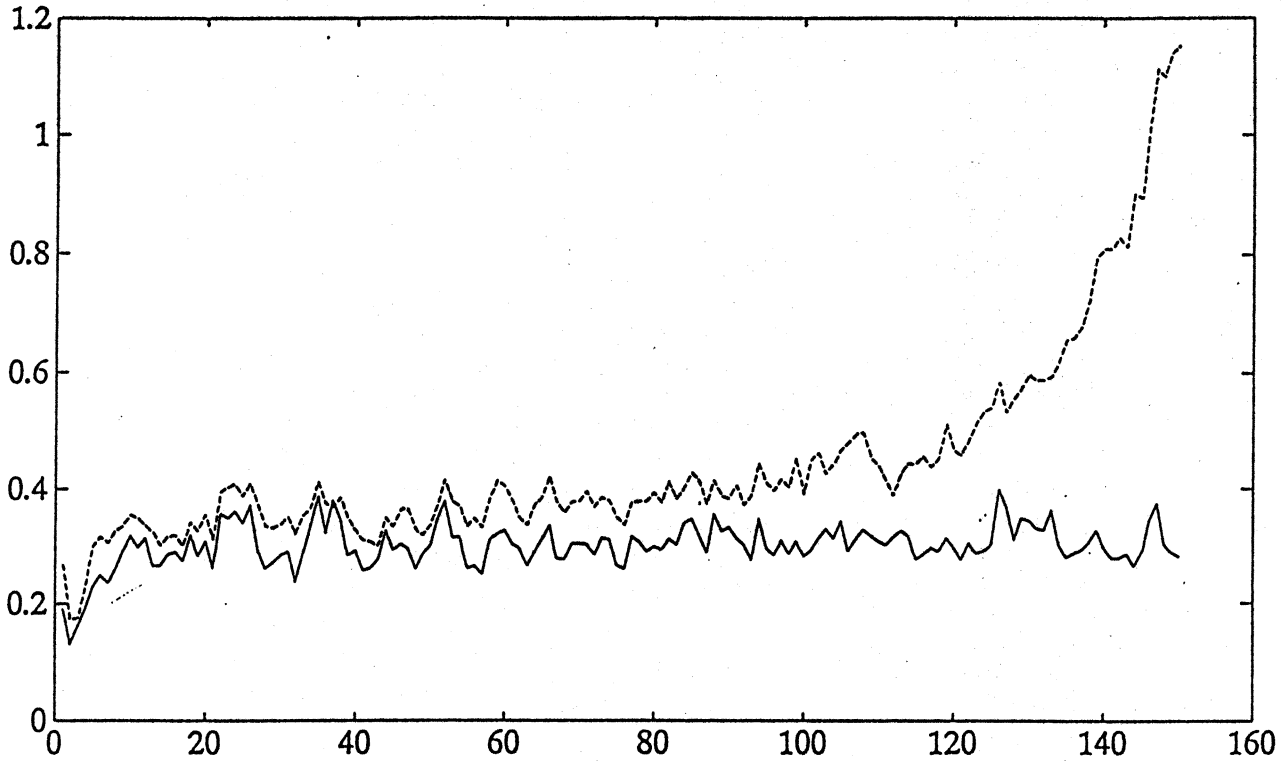

Fig. 20. RMSEs versus time of NFWA $(—)$ and NF$(---)$ for Example 5.

ferred to as an NF, were compared. The RMSEs averaged over 500 runs of the NFWA $(—)$ and the NF $(---)$ are plotted versus time in Fig. 20. The length of each training data sequence is 100. The recursive neural filters were tested for 150 time points to assess their ability to generalize. The NFWA has a better performance than the NF, before 100 time points. After 100 time points, the latter deteriorates rapidly while the former remains good.

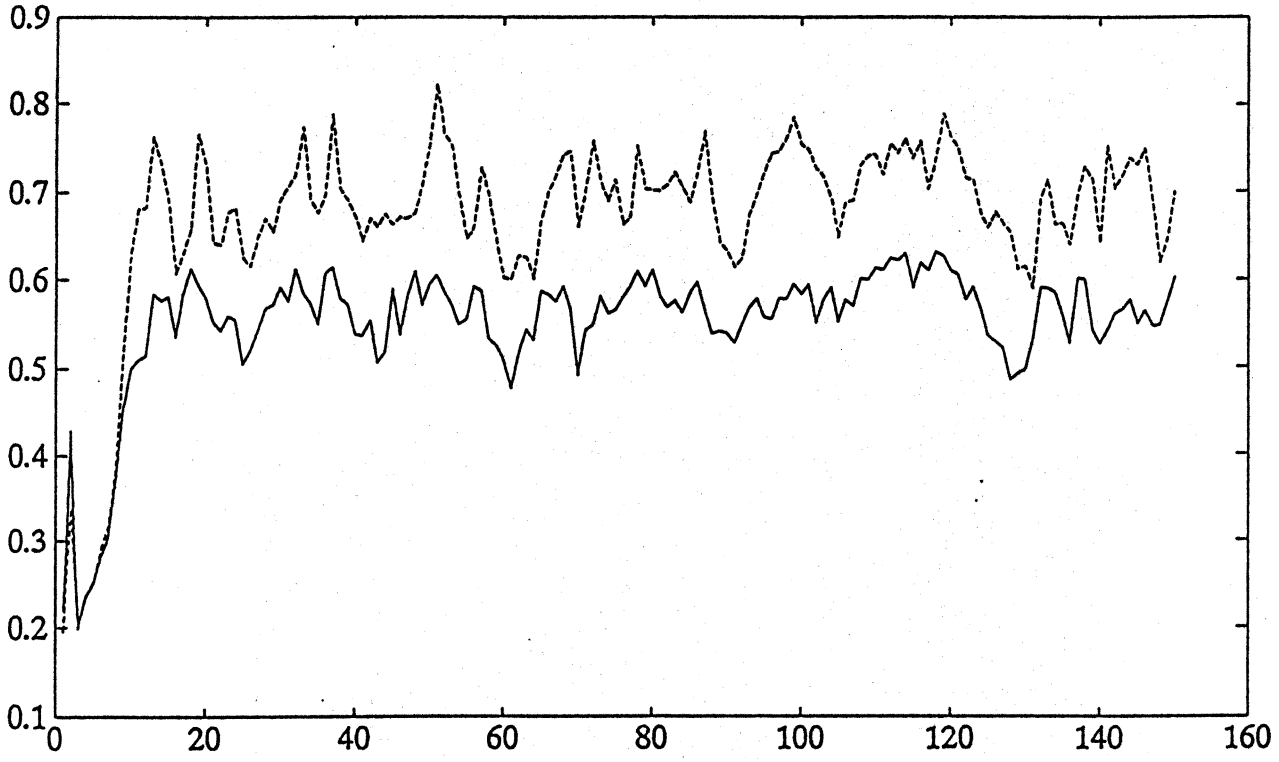

Fig. 21. RMSEs of neural filter (—) and EKF (– – –) versus time for Example 6.

Example 6: The signal and measurement processes are

$$x_{n+1} = x_n + 1.2\sin x_n + 1.21 + 0.2\xi_n \tag{17}$$

$$y_n = x_n + \varepsilon_n \tag{18}$$

where ξ_n and ε_n are independent standard white Gaussian sequences, and $x_0 = 0$. Notice that both the signal and measurement processes grow over time.

The EKF tracks the signal process reasonably well. However, a recursive neural filter using sigmoidal scaling functions to, respectively, reduce and extend the measurements and outputs of the RNN in the filter fails to work. A recursive neural filter consisting of an MLPWIN and a range reducer by differencing and a range extender by Kalman filtering is synthesized and compared with an EKF. The MLPWIN has a single hidden layer of nine fully interconnected nodes. The length of each training data sequence is 100. The recursive neural filter is tested for 150 time points to assess its generalization capability. The RMSEs averaged over 500 runs are plotted versus time for the NF(—) and the EKF (– – –) in Fig. 21. Note that the RMSE difference between the NF and EKF is caused by omitting nonlinearity in linearization for deriving the EKF.

IX. Conclusion

The synthetic approach to optimal filtering is simple, general, systematic, and practical. While the conventional analytic approach to optimal filtering derives formulas or equations from a mathematical model of the signal and measurement processes, the synthetic approach synthesizes realizations of those processes into a virtually optimal filter. Such realizations are obtained by computer simulations or actual experiments. The synthetic approach has the following advantages.

1) No such assumption as the Markov property, linear dynamics, Gaussian distribution, or additive noise is necessary.
2) It applies even if a mathematical model of the signal or measurement process is not available.
3) The resulting recursive neural filter is virtually minimum variance for its given architecture.
4) Recursive neural filters with or without dynamical range transformers are parsimonous universal approximators of optimal recursive filters.
5) Other estimation error criteria than the minimum-variance criterion can be easily used.
6) The EKF and a recursive neural network can be easily integrated into a neural filter.
7) Recursive neural filters synthesized offline do not perform Monte Carlo online and, therefore, require much less online computation than filters doing online Monte Carlo.
8) Recursive neural filters are well suited for real-time processing due to their massively parallel nature of computing.
9) The simple recursive neural network architecture is suitable for chip implementation.

The synthetic approach was first reported in [17] and [19], in which MLPWINs are used as recursive neural filters. In this paper, MLPWOFs are proposed instead. If the ranges of the signal and measurement processes are compact, the MLPWOFs are universal approximators of optimal filters.

An MLPWOF used as a recursive neural filter allows us to load an estimate of the initial signal into it, improving the transient state and hastening the convergence of the filter.

If the range of the signal or measurement process expands in time or is very large relative to the resolution or accuracy of the filtering required, dynamical range reducers or extenders, respectively, are proposed in this paper. Dynamical range reducers and extenders have the following advantages.

1) They, respectively, reduce the input and output range required of an RNN.
2) They reduce the amount of training data required.
3) They reduce the RNN size and training difficulty for a given filtering resolution or accuracy.
4) They improve the generalization ability of the recursive neural filter, especially beyond the length of the period over which the training data are collected.

If a dynamical range reducer is needed and the signal is nonobservable from the outputs of the dynamical range reducer, loading an estimate of the initial signal into a recursive neural filter can overcome this nonobservability problem. In this case, an MLPWOF is needed.

Consider using an MLPWIN and an MLPWOF both with a single hidden layer of nodes for optimal filtering for the same signal and measurement processes. To determine the architecture of the latter, we need to determine the number of nodes in the hidden layer and the number of free feedbacks. On the other hand, to determine the architecture of the former, we need to determine only the number of nodes in the hidden layer. This is certainly an advantage in synthesizing a filter. However, an estimate of the initial signal cannot be loaded into an MLPWIN. Therefore, an MLPWIN with its outputs, which are all teacher-influenced, feedbacked after a unit-time delay to the input layer may be a more desirable architecture than that of an MLPWIN or MLPWOF. In such an RNN with teacher-influenced feedbacks, only the number of nodes in the single hidden layer needs to be determined, and an estimate of the initial signal can be loaded into the RNN. Such RNNs are obviously universal approximators of the optimal filters if the ranges of the signal and measurement processes are compact. Numerical examples showing the numerical feasibility of such RNNs as recursive neural filters will soon be reported.

The development of the method of global minimization through convexification [20]–[22] has greatly alleviated the local-minimum problem in training neural networks, which is the main difficulty in the synthetic approach.

REFERENCES

[1] R. E. Kalman, "A new approach to linear filtering and prediction problems," *J. Basic Eng. Trans. ASME*, vol. D82-1, pp. 35–45, 1960.

[2] R. E. Kalman and R. S. Bucy, "New results in linear filtering and prediction theory," *J. Basic Eng. Trans. ASME*, vol. D83-3, pp. 95–108, 1961.

[3] R. S. Liptser and A. N. Shiryayev, *Statistics of Random Processes I: General Theory*. New York: Springer-Verlag, 1977.

[4] R. S. Bucy and P. D. Joseph, *Filtering for Stochastic Processes with Applications to Guidanc*, 2nd ed. New York: Chelsea, 1987.

[5] G. Kallianpur and R. L. Karandikar, *White Noise Theory of Prediction, Filtering, and Smoothing*. New York: Gordon & Breach, 1988.

[6] J. T. Lo, "Finite dimensional sensor orbits and optimal nonlinear filtering," Ph.D. dissertation, Univ. Southern California, Los Angeles, 1969.

[7] ——, "Finite dimensional sensor orbits and optimal nonlinear filtering," *IEEE Trans. Inform. Theory*, vol. IT-18, pp. 583–588, Sept. 1972.

[8] ——, "Exponential Fourier densities and estimation and detection on a circle," *IEEE Trans. Inform. Theory*, vol. IT-23, pp. 110–116, Jan. 1977.

[9] J. T. Lo and L. R. Eshleman, "Exponential fourier densities on s^2 and optimal estimation and detection for directional processes," *IEEE Trans. Inform. Theory*, vol. IT-23, pp. 321–336, May 1977.

[10] ——, "Exponential Fourier densities on $S0(3)$ and optimal estimation and detection for rotational processes," *SIAM J. Appl. Math.*, vol. 36, no. 1, pp. 73–82, Feb. 1979.

[11] ——, "Exponential Fourier densities and optimal estimation for axial processes," *IEEE Trans. Inform. Theory*, vol. IT-25, pp. 463–470, July 1979.

[12] S. K. Mitter and D. Ocone, "Multiple integral expansions for nonlinear filtering," presented at the 18th IEEE Conf. Decision and Control, Ft. Lauderdale, FL, 1979.

[13] J. T. Lo and S. K. Ng, "Optimal orthogonal expansion for estimtion I: Signal in white Gaussian noise," in *Nonlinear Stochastic Problems*, R. S. Bucy, J. M. F. Moura, and D. Reidel, Eds. Dordrecht, The Netherlands, 1983, pp. 291–310.

[14] J. T. Lo and S.-K. Ng, "Optimal Fourier-hermite expansion for estimation," *Stoch. Process. Appl.*, vol. 21, pp. 291–304, 1986.

[15] J. T. Lo, "Optimal estimation for the satellite attitude using star tracker measurements," *Automatica*, vol. 22, pp. 477–482, 1986.

[16] J. T. Lo and S.-K. Ng, "Optimal functional expansion for estimation from counting observations," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 21–35, Jan. 1987.

[17] J. T. Lo, "Synthetic approach to optimal filtering," in *Proc. 1992 Int. Simulation Technology Conf. and 1992 Workshop Neural Networks*, pp. 475–481.

[18] ——, "Optimal filtering by recurrent neural networks," presented at the 30th Annu. Allerton Conf. Communication, Control and Computing, Monticello, IL, 1992.

[19] ——, "Synthetic approach to optimal filtering," *IEEE Trans. Neural Networks*, vol. 5, pp. 803–811, Sept. 1994.

[20] ——, "Minimization through convexitization in training neural networks," in *Proc. 2002 Int. Joint Conf. Neural Networks*, 2002, pp. 1558–1563.

[21] J. T. Lo and D. Bassu, "Robust identification of dynamic systems by neurocomputing," presented at the 2001 Int. Joint Conf. Neural Networks, Washington, DC.

[22] ——, "An adaptive method of training multilayer perceptrons," presented at the 2001 Int. Joint Conf. Neural Networks, Washington, D.C..

[23] J. T. Lo and L. Yu, "Overcoming recurrent neural networks' compactness limitation for neurofiltering," in *Proc. 1997 Int. Conf. Neural Networks*, pp. 2181–2186.

[24] J. T. Lo, "Neural network approach to optimal filtering," Rome Lab., Air Force Material Command, Rome, NY, Tech. Rep. RL-TR-94 197, 1994.

[25] T. Parisini and R. Zoppoli, "Neural networks for nonlinear state estimation," *Int. J. Robust Nonlinear Control*, vol. 4, pp. 231–248, 1994.

[26] A. Alessandri, M. Baglietto, T. Parisini, and R. Zoppoli, "A neural state estimator with bounded errors for nonlinear systems," *IEEE Trans. Automat. Contr.*, vol. 44, pp. 2028–2042, Nov. 1999.

[27] A. Alessandri, T. Parisini, and R. Zoppoli, "Neural approximators for nonlinear finite-memory state estimation," *Int. J. Control*, vol. 67, pp. 275–301, 1997.

[28] T. Parisini, A. Alessandri, M. Maggiore, and R. Zoppoli, "On convergence of neural approximate nonlinear state estimators," in *Proc. 1997 Amer. Conf. Control*, vol. 3, pp. 1819–1822.

[29] S. C. Stubberud, R. N. Lobbia, and M. Owen, "An adaptive extended Kalman filter using artificial neural networks," in *Proc. 34th IEEE Conf. Decision and Control*, 1995, pp. 1852–1856.

[30] ——, "Targeted on-line modeling for an extended Kalman filter using artificial neural networks," in *Proc. 1998 IEEE Int. Joint Conf. Neural Networks*, vol. 2, pp. 1019–1023.

[31] A. G. Parlos, S. K. Menon, and A. F. Atiya, "An algorithmic approach to adaptive state filtering using recurrent neural networks," *IEEE Trans. Neural Networks*, vol. 12, pp. 1411–1432, Nov. 2001.
[32] S. Elanayar and Y. C. Shi, "Radial basis function neural networks for approximation and estimation of nonlinear stochastic dynamic systems," *IEEE Trans. Neural Networks*, vol. 5, pp. 594–603, July 1994.
[33] S. Haykin, P. Yee, and E. Derbez, "Optimum nonlinear filtering," *IEEE Trans. Signal Processing*, vol. 45, pp. 2774–2786, Nov. 1997.
[34] A. Doucet, N. de Freitas, and N. Gordon, *Sequential Monte Carlo Methods in Practice*. New York: Springer-Verlag, 2001.
[35] S. J. Julier, J. K. Uhlmann, and H. F. Durrant-Whyte, "A new approach for the nonlinear transformation of means and covariances in filters and estimators," *IEEE Trans. Automat. Contr.*, vol. 45, pp. 477–482, Mar. 2000.
[36] S. J. Julier and J. K. Uhlmann, "Data fusion in nonlinear systems," in *Handbook of Data Fusion*, D. Hall and J. Llinas, Eds. Boca Raton, FL: CRC, 2001, ch. 3.
[37] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, pp. 359–366, 1989.
[38] S. Haykin, *Adaptive Filter Theory*, 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1996.
[39] D. T. M. Slock and T. Kailath, "Numerically stable fast transversal filters for recursive least squares adaptive filtering," *IEEE Trans. Signal Processing*, vol. 39, pp. 92–114, Jan. 1991.

James T. Lo received the B.A. degree in mechanical engineering from National Taiwan University, Taipei, R.O.C., and the Ph.D. degree in aerospace engineering from the University of Southern California, Los Angeles.

He was a Postdoctoral Research Associate at Stanford University, Stanford, CA, and at Harvard University, Cambridge, MA. He is currently a Professor in the Department of Mathematics and Statistics at University of Maryland Baltimore County, Baltimore. His research interests have included optimal filtering, signal processing, and neural computing. In 1992, he published a synthetic approach to optimal filtering using recurrent neural networks and obtained a best paper award for it. The approach is the first practical solution of the long-standing optimal nonlinear filtering problem in a most general setting. The recursive neural filter has been proven to approximate the optimal nonlinear filter in performance to any accuracy. In recent years, he has developed general and systematic methodologies for adaptive, accommodative, and/or robust processing, their applications to active noise/vibration control, an effective convexification method of overcoming the local-minimum problem in training neural networks and estimating nonlinear regression models, and a novel approach to pattern classification and data compression.

Lei Yu received the B.S. degree in geophysics from the Chinese University of Science and Technology, Hefei, China, in 1987 and the Ph.D. degree in physics from University of Maryland, College Park, in 1992.

He was a Senior Research Scientist at the Maryland Technology Corporation. He is currently a Senior Consultant at Microsoft Corporation, Washington, DC. His research interests include neural networks and signal processing.