A Fast U-D Factorization-Based Learning Algorithm with Applications to Nonlinear System Modeling and Identification

Youmin Zhang and X. Rong Li

***Abstract*— A fast learning algorithm for training multilayer feedforward neural networks (FNN's) by using a fading memory extended Kalman filter (FMEKF) is presented first, along with a technique using a self-adjusting time-varying forgetting factor. Then a U-D factorization-based FMEKF is proposed to further improve the learning rate and accuracy of the FNN. In comparison with the backpropagation (BP) and existing EKF-based learning algorithms, the proposed U-D factorization-based FMEKF algorithm provides much more accurate learning results, using fewer hidden nodes. It has improved convergence rate and numerical stability (robustness). In addition, it is less sensitive to start-up parameters (e.g., initial weights and covariance matrix) and the randomness in the observed data. It also has good generalization ability and needs less training time to achieve a specified learning accuracy. Simulation results in modeling and identification of nonlinear dynamic systems are given to show the effectiveness and efficiency of the proposed algorithm.**

***Index Terms*—BP algorithm, extended Kalman filter, feedforward neural networks, forgetting factor, U-D factorization.**

I. Introduction

THE CLASSICAL method for training a multilayer feedforward neural network (FNN) is the steepest descent backpropagation (BP) algorithm. The BP algorithm suffers from a number of shortcomings, including a slow learning rate. A number of learning algorithms have been proposed in an attempt to speed up the learning rate. Among them, the one based on the extended Kalman filter (EKF) technique has received considerable attention recently. It has been recognized that the EKF can provide a substantial speed-up in training time at the expense of a higher computational cost at each time step, compared with other simpler on-line gradient-based algorithms. Singhal and Wu [10] was the first to suggest that the training of a multilayer FNN could be interpreted as an estimation problem for a nonlinear dynamic system that can be solved by using the EKF algorithm. More recently, a number of researchers have investigated EKF-based learning algorithms and extended them to several network structures [3], [4], [7]–[9], [12]. It was assumed in [8] that groups of weights exist which are mutually uncorrelated. This assumption leads to a block structure for the error covariance matrix. A decoupled EKF was then proposed to reduce the computational complexity, while leaving the numerical instability problem unsolved. Two other somewhat similar algorithms were given in [4] and [9]. These algorithms improved the convergence rate considerably and exhibited good performance. However, their numerical stability is not guaranteed. This may degrade convergence and increase training time.

Manuscript received May 13, 1996; revised January 6, 1998 and January 19, 1999. This work was supported in part by ONR under Grant N0014–97–1–0570, NSF under Grants ECS-9409358 and ECS-9734285, LEQSF under Grant 1996-99-RD-A-32, NNSFC under Grant 69274015, and ASFC under Grant 94E53186.

Y. Zhang is with the Department of Electrical and Computer Engineering, The University of Western Ontario, London, Ontario N6A 5B9, Canada, on leave from the Department of Automatic Control, Northwestern Polytechnical University, Xian, Shaanxi 710072, China.

X. Rong Li is with the Department of Electrical Engineering, University of New Orleans, New Orleans, LA 70148 USA.

Publisher Item Identifier S 1045-9227(99)05996-2.

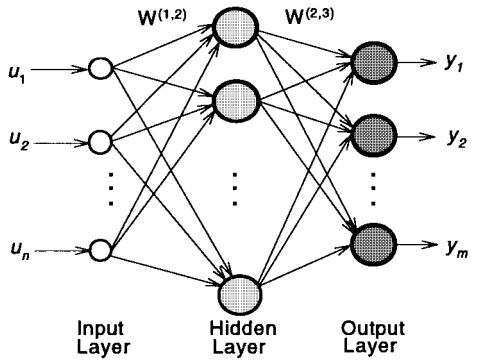

Fig. 1. A feedforward neural network.

In this paper, a technique based on a self-adjusting time-varying forgetting factor, combined with an EKF, for training FNN's is presented. Next, a much faster and more robust implementation of the learning algorithm is proposed using the forgetting factor technique and a U-D factorization[1] of the EKF. This proposed algorithm is superior to the BP algorithms because it has improved numerical stability, convergence, accuracy, and computational complexity.

II. Fading Memory Extended Kalman Filter-Based Learning Algorithm

FNN's have been the subject of intensive research efforts in recent years because of their interesting learning and generalization capability and applicability in a variety of classification, approximation, modeling, identification, and control problems. Fig. 1 shows a typical structure of a single hidden-layer FNN.

A. EKF Formulation of FNN Learning

The above FNN can be trained by adjusting its weights using a stream of input–output observations $\{\mathbf{u}(t), \mathbf{y}(t)\colon t = 1, \cdots, N\}$, where N is the number of training data samples. The objective is to obtain a set of weights $\{W^{(1,2)}, W^{(2,3)}\}$ such that the neural network predicts future outputs accurately. Although this training mechanism is generally referred to as

[1]This is a matrix factorization method, in which the covariance matrix P is decomposed into $P = UDU^T$ such that U is unit upper triangular and D is diagonal [1].

supervised learning, it can be also considered as a nonlinear estimation problem where the weight values are unknown and to be estimated for the given set of inputs and outputs. An approach is to concatenate all the network parameters into a state vector $\boldsymbol{\theta}(t)$ and define an operator $\mathbf{f}(\cdot)$ to perform the function of an FNN that maps the state (parameter) vector and the input onto the output. The concatenated state vector $\boldsymbol{\theta} \in \Re^{n_\theta}$ (n_θ is the total number of the weights and thresholds) can be defined as

$$\begin{aligned}\boldsymbol{\theta} &= [W_{11}^{(1,2)}, \cdots, W_{n_2 n_1}^{(1,2)}, b_1, \cdots, b_{n_2}, W_{11}^{(2,3)}, \cdots, W_{n_3 n_2}^{(2,3)}]^T \\ &= [\theta_1, \theta_2, \cdots, \theta_{n_\theta}]^T \end{aligned} \tag{1}$$

where $n_\theta = (n_1 + 1)n_2 + n_2 n_3$; $n_1 = n$, $n_2 = n_h$, $n_3 = m$. Then, the training of an FNN can be posed as a state estimation problem with the following dynamic and observation equations for the neural network:

$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) \tag{2}$$

$$\mathbf{y}(t) = \mathbf{f}[\boldsymbol{\theta}(t), \mathbf{u}(t)] + \boldsymbol{\varepsilon}(t) \tag{3}$$

where $\boldsymbol{\theta}(t)$ is the state of the FNN at time t, $\mathbf{u}(t)$ is the input of the FNN; $\mathbf{y}(t)$ is the observed (or desired) output of the FNN; $\boldsymbol{\varepsilon}(t)$ is the measurement noise, which may also include modeling errors of the FNN. It is assumed white and Gaussian with zero mean and covariance matrix $R(t)$. Note that (2) is the state transition of $\boldsymbol{\theta}(t)$ from t to $t+1$, with the transition matrix equal to an identity matrix.

The state estimation is then the problem of determining $\hat{\boldsymbol{\theta}}$ that minimizes the sum of squared prediction errors of all prior observations embedded in the function

$$J = \tfrac{1}{2} \sum_{t=1}^{N} \{\mathbf{y}(t) - \mathbf{f}[\hat{\boldsymbol{\theta}}(t), \mathbf{u}(t)]\}^2 \lambda^{N-t}(t) \tag{4}$$

where $\lambda(t)$ is a time-varying forgetting factor. By a first-order Taylor series expansion of $\mathbf{f}[\boldsymbol{\theta}(t), \mathbf{u}(t)]$ about an estimated state $\hat{\boldsymbol{\theta}}(t)$, the estimated output $\hat{\mathbf{y}}(t)$ can be obtained by the following linearized equation:

$$\hat{\mathbf{y}}(t) = \mathbf{f}[\hat{\boldsymbol{\theta}}(t), \mathbf{u}(t)] + H(t)[\boldsymbol{\theta}(t) - \hat{\boldsymbol{\theta}}(t)] + \text{HOT} \tag{5}$$

where higher order terms HOT may be neglected or included in the measurement noise. $H(t)$ is the Jacobian matrix, that is, the partial derivative of the nonlinear function $\mathbf{f}[\boldsymbol{\theta}(t), \mathbf{u}(t)]$ with respect to $\boldsymbol{\theta}$ evaluated at the best possible state estimate available

$$\begin{aligned} H(t) &= \left.\frac{\partial \mathbf{f}[\boldsymbol{\theta}(t), \mathbf{u}(t)]}{\partial \boldsymbol{\theta}(t)}\right|_{\boldsymbol{\theta}(t) = \hat{\boldsymbol{\theta}}(t-1)} \\ &= \left[\frac{\partial \mathbf{f}}{\partial \mathbf{W}^{(1,2)}} \frac{\partial \mathbf{f}}{\partial \mathbf{b}} \frac{\partial \mathbf{f}}{\partial \mathbf{W}^{(2,3)}}\right]^T_{\boldsymbol{\theta}(t) = \hat{\boldsymbol{\theta}}(t-1)}. \end{aligned}$$

Then a fading-memory EKF (FMEKF) is given by

$$\hat{\boldsymbol{\theta}}(t) = \hat{\boldsymbol{\theta}}(t-1) + K(t)\{\mathbf{y}(t) - \mathbf{f}[\hat{\boldsymbol{\theta}}(t-1), \mathbf{u}(t-1)]\} \tag{6}$$

$$\begin{aligned} K(t) = {} & [H(t)P(t-1)]^T \\ & \cdot \{[H(t)P(t-1)]H^T(t) + \lambda(t)R(t)\}^{-1} \end{aligned} \tag{7}$$

$$P(t) = \{P(t-1) - K(t)[H(t)P(t-1)]\}/\lambda(t) \tag{8}$$

where $P(t)$ is the error covariance matrix; $K(t)$ is the filter gain.

Two designs of time-varying forgetting factors are considered in this paper.

1) Assume $\lambda(t) = 1$ in (7) and (8), and choose the noise covariance matrix $R(t) = e^{-t/N}R$ instead of R, where R is a constant.
2) Set $R(t) = I$ in (7) and assume a time-varying forgetting factor as follows [5]:

$$\lambda(t) = \lambda_0 \lambda(t-1) + (1 - \lambda_0) \tag{9}$$

where the variation rate λ_0 and the initial forgetting factor $\lambda(0)$ are design parameters. A set of typical values is $\lambda_0 = 0.99$ and $\lambda(0) = 0.95$.

It should be pointed out that the order of an EKF used as an FNN training algorithm grows with the number of the network weights. In each cycle, its computation is heavier than that of a BP-type training algorithm. However, the matrix to be inverted in (7) is $m \times m$, where in general $m \ll n_\theta$. In the single-output case, the matrix inversion reduces to a scalar division. Note also that the factor $H(t)P(t-1)$ occurs three times in the computation of $K(t)$ and $P(t)$. This has been utilized in the implementation of the above FMEKF to reduce computational requirement. Another effective way to reduce the computation requirement and at the same time enhance the numerical robustness is to use the so-called U-D factorization filter, to be presented in the next section. Moreover, the above FMEKF actually reduces to the weighted recursive least squares (RLS) identification due to the fact that the state transition matrix of $\boldsymbol{\theta}(t)$ is an identity matrix.

B. Self-Adjustment of Time-Varying Forgetting Factor

Forgetting factors are often used for tracking (slowly) varying dynamic systems in recursive identification and adaptive control algorithms [5]. They can also be used to prevent the Kalman filter from divergence due to model mismatch or nonlinearity in the system dynamics. In some cases, however, for example in adaptive control when the process is at steady state, there is little change in the input and output. In such a case, if the identification were allowed to continue, problems such as covariance "wind-up" or "blow-up" could occur. This is similar to the training of an FNN discussed in this paper. When the training process reaches a specified accuracy, the current input and output data should not have greater weights than other data and, therefore, the forgetting factor should be made adaptive and held as one at the steady state to allow a fine adjustment of the weights. On the other hand, if the specified accuracy is not reached in the subsequent training iterations, the time-varying forgetting factor should be in effect to speed up convergence. That is why a time-varying forgetting factor has been used in some identification and adaptive control algorithms and also in this paper. In this way, both a fast convergence and a high learning accuracy can be achieved simultaneously. Here, the use of a nonunity forgetting factor is to speed up the learning rate at the initial stage, whereas in the later stages it is set to unity to avoid possible fluctuations of learning process and to ensure a smoothing convergence

to the minimum value. The adaptive adjustment process was implemented by a logic based on the detection of a "large" training error, described next.

The root mean square errors (RMSE's) between the estimated and the actual output of the FNN is a natural measure of the accuracy of the training algorithm, which is defined as

$$\text{RMSE} = \left(\frac{1}{N} \sum_{i=1}^{N} \boldsymbol{\epsilon}_i^2(t) \right)^{1/2} \tag{10}$$

where N denotes the number of time points in a training sample. The RMSE was used as an index in the proposed adaptive adjustment of the forgetting factor for each iteration of the training process. When it is larger than some specified learning accuracy δ (e.g., $\delta = 10^{-4}$), $\lambda(t)$ was set a value smaller than but close to one, as computed by (9); otherwise, it was set to one. That is

$$\lambda(t) = \begin{cases} \lambda_0 \lambda(t-1) + (1-\lambda_0), & \text{if RMSE} > \delta \\ 1, & \text{if RMSE} \le \delta. \end{cases} \tag{11}$$

The threshold δ should be chosen to have a balanced training accuracy and smoothness of training process.

A similar rule was followed in the first design of the forgetting factor.

III. U-D Factorization-Based FMEKF Learning Algorithm

The computation of the covariance matrix $P(t)$ in the FMEKF plays an important role. Due to truncation and rounding off errors in a computer, the algorithm may lead to the loss of symmetry and positive definiteness of $P(t)$, or possibly divergence. Several matrix factorization methods, such as square-root, U-D factorization, and singular value decomposition (SVD), have been developed to improve the computation of $P(t)$. Among them the U-D factorization method is one of the most popular ones due to its computational efficiency. This method guarantees the positive-definiteness and symmetry of $P(t)$, and thus high estimation accuracy and robustness can be attained [1], [5].

In order to develop the U-D factorization-based learning algorithm, we first give results for the case of multiple inputs single output (MISO), then extend it to the case of multiple inputs multiple outputs (MIMO).

A. Multiple-Input Single-Output (MISO) Case

In order to carry out the U-D decomposition for $P(t)$, we first substitute (7) into (8) to get

$$P(t) = \frac{1}{\lambda(t)} \cdot \left\{ P(t-1) - \frac{P(t-1)H^T(t)H(t)P(t-1)}{H(t)P(t-1)H^T(t) + \lambda(t)R(t)} \right\}. \tag{12}$$

Suppose $P(t-1)$ has been U-D decomposed as $P(t-1) = U(t-1)D(t-1)U^T(t-1)$ at time $t-1$, (12) then becomes

$$P(t) = \frac{1}{\lambda(t)} U(t-1) \left[D(t-1) - \frac{\mathbf{g}\mathbf{g}^T}{\beta} \right] U^T(t-1) \tag{13}$$

where

$$\mathbf{g} = D(t-1)\mathbf{e}, \quad \mathbf{e} = U^T(t-1)H^T(t) \tag{14}$$

$$\beta = \mathbf{e}^T D(t-1)\mathbf{e} + \lambda(t)R(t) \tag{15}$$

and $H(t)$, $\mathbf{e}$ and $\mathbf{g}$ are n_θ-vectors for an FNN with a single output; and $R(t)$ and β are scalars.

Note that since the product of two unit upper triangular matrices is still a unit upper triangular matrix, to obtain U-D factors of $P(t)$ it is sufficient to obtain U-D factors for the bracketed matrix in (13). In fact, if $\overline{U}(t-1)$ and $\overline{D}(t-1)$ are U-D factors of the matrix $[D(t-1) - (\mathbf{g}\mathbf{g}^T/\beta)]$, then the U-D factors of $P(t) = U(t)D(t)U^T(t)$ are

$$U(t) = U(t-1)\overline{U}(t-1); \quad D(t) = \overline{D}(t-1)/\lambda(t) \tag{16}$$

where a detailed derivation of $\overline{U}(t-1)$ and $\overline{D}(t-1)$ can be found in [1] and [5].

Let e_i and g_i, $i = 1, \cdots, n_\theta$, be the elements of $\mathbf{e}$ and $\mathbf{g}$, respectively. Let $U_{ij}(t)$ and $D_{ij}(t)$ represent the (i,j)th element of $U(t)$ and $D(t)$, respectively. Then, the above FMEKF (6)–(8) can be implemented in U-D factorization form as follows.

Step 1) Compute $\mathbf{e} = [e_1, \cdots, e_{n_\theta}]^T := U^T(t-1)H^T(t)$, $\mathbf{g} = [g_1, \cdots, g_{n_\theta}]^T := D(t-1)\mathbf{e}$, $\alpha_0 := \lambda(t)R(t)$.

Step 2) For $j = 1, \cdots, n_\theta$, compute

$$\begin{cases} \alpha_j := \alpha_{j-1} + e_j g_j \\ D_{jj}(t) := D_{jj}(t-1)\alpha_{j-1}/\alpha_j \lambda(t) \\ \nu_j := g_j \\ \mu_j := -e_j/\alpha_{j-1}. \end{cases} \tag{17}$$

For $i = 1, \cdots, j-1$, compute

$$\begin{cases} U_{ij}(t) := U_{ij}(t-1) + \nu_i \mu_j \\ \nu_i := \nu_i + U_{ij}(t-1)\nu_j. \end{cases} \tag{18}$$

Step 3) Compute gain

$$K(t) = [\nu_1, \cdots, \nu_{n_\theta}]^T / \alpha_{n_\theta}. \tag{19}$$

Step 4) Compute

$$\hat{\boldsymbol{\theta}}(t) = \hat{\boldsymbol{\theta}}(t-1) + K(t) \cdot \{ y(t) - f[\hat{\boldsymbol{\theta}}(t-1), \mathbf{u}(t-1)] \}. \tag{20}$$

B. Multiple-Input Multiple-Output (MIMO) Case

For MIMO systems, one approach is to transform the problem into a sequence of scalar problems [1], [5]. Based on this idea, a U-D factorization filter for multiple outputs can be obtained by processing the m-dimensional output with m sequential applications of the above single-output U-D factorization based FMEKF. That is, the vector equation (3) for the desired observation vector $\mathbf{y}(t)$ is equivalent to the following scalar equations:

$$y_k(t) = f_k[\boldsymbol{\theta}(t), \mathbf{u}(t)] + \varepsilon_k(t), \qquad k = 1, \cdots, m. \tag{21}$$

The resulting algorithm is similar to the one for the MISO case, except that in Step 2, e_j and g_j are replaced by e_{jk} and g_{jk}. The calculation of $K_k(t)$ in Step 3) is now given by a $K_k(t) = \nu_j/\alpha_{n_\theta}$, and the final gain matrix is given by $n_\theta \times m$ dimensional matrix, $K(t) = [K_1(t) \cdots K_m(t)]$.

TABLE I
RMSE VERSUS TRAINING ITERATIONS FOR DIFFERENT ALGORITHMS

Iterations	BP	EKF	FMEKF		UD-EKF	UD-FMEKF	
k	$\eta_w = 0.2$		Design 1	Design 2		Design 1	Design 2
10	4.4526E-2	3.3998E-2	3.3612E-2	1.2714E-3	3.6583E-2	3.2333E-2	1.0536E-4
50	2.0233E-2	9.0001E-3	5.3066E-3	1.1026E-4	8.2387E-3	5.1093E-3	6.6708E-6
100	2.4396E-2	5.3425E-3	1.8112E-3	7.9239E-5	4.5838E-3	1.7367E-3	2.6899E-6
150	1.2114E-2	3.9453E-3	8.2368E-4	6.9968E-5	3.3810E-3	7.8098E-4	9.1077E-7
200	1.0505E-2	3.1955E-3	4.5404E-4	1.8234E-4	2.5978E-3	4.1825E-4	8.9983E-7

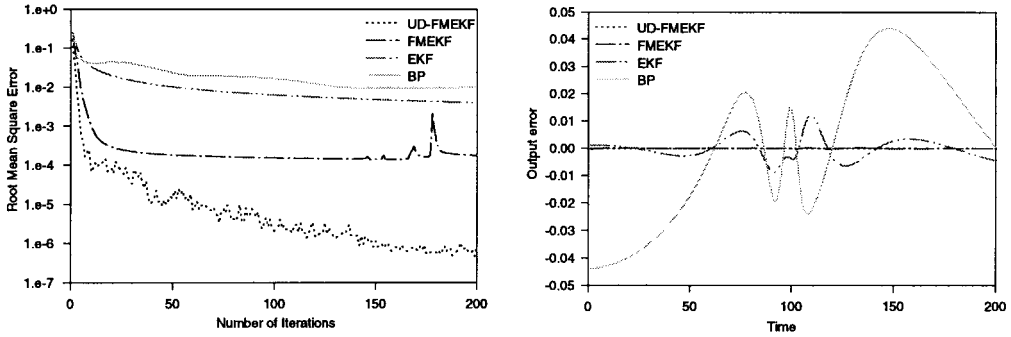

Fig. 2. RMSE and output errors of four algorithms for example 1.

IV. APPLICATIONS TO NONLINEAR SYSTEMS MODELING AND IDENTIFICATION

In this section, simulation results for several nonlinear plant identification problems are presented. A performance comparison among the UD-FMEKF, FMEKF, EKF, and BP algorithms is given and analyzed in terms of learning accuracy, convergence rate, influence of initial weights and covariance, influence of noise, and the generalization ability of the networks.

1) Example 1: Consider the problem of modeling and identification of a nonlinear static system

$$y(t) = 2.0\frac{x(t)}{1+x^2(t)} \tag{22}$$

where $y(t) \in [-1, 1]$ when $x(t) \in [-10, 10]$. 200 input data points for $x(t)$ were generated uniformly in the range of $[-10, 10]$. The UD-FMEKF, FMEKF, EKF, and BP algorithms were employed to train an FNN, with an architecture of 1–10–1 consisting of a total of 30 weights, to approximate (22).

2) Identification Accuracy: Table I shows the RMS error (RMSE) of the different algorithms versus the number of iterations. Designs 1 and 2 correspond to the first and second designs of the forgetting factor, respectively. Fig. 2 illustrates the RMSE and output error curves of the four algorithms. It is clear that the learning accuracy and convergence of the proposed UD-FMEKF algorithm is much better than those of the FMEKF, EKF, and BP algorithms, especially for Design 2 of the forgetting factor. The FMEKF algorithm also has much better accuracy and convergence rate than the EKF and BP algorithms. The important role of the proposed forgetting factor technique can be seen clearly from these results. The learning rate and momentum constant for the BP algorithm were selected as $\eta_w = 0.2$ and $\rho_w = 0.2$ because larger values would probably lead to an oscillation or even divergence of the BP algorithm and smaller values would result in a slower convergence.

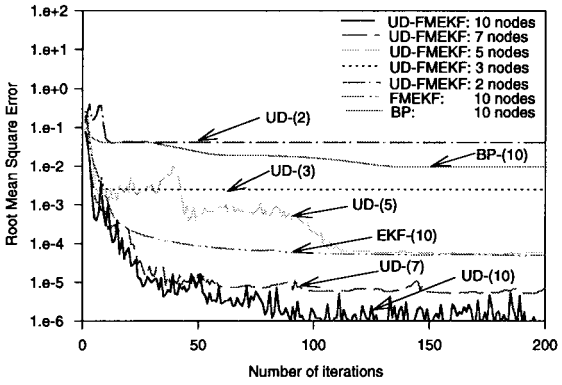

Fig. 3. Performance comparison of different hidden nodes.

3) The Influence of the Number of Hidden Nodes: Fig. 3 illustrates the RMSE curves of the FNN's with different numbers of hidden nodes trained by the above algorithms. It is clear that 1) similar learning accuracies and convergence rates were obtained by the UD-FMEKF with just five nodes and by the FMEKF with ten nodes; 2) the UD-FMEKF with only three nodes had a much better learning accuracy and convergence rate than the BP algorithm with ten nodes; and 3) the FMEKF had a much better learning accuracy and convergence rate than BP. This demonstrates the superiority of the UD-FMEKF algorithm. It also indicates that to obtain a given modeling accuracy, fewer hidden nodes and less computation time are needed for the UD-FMEKF.

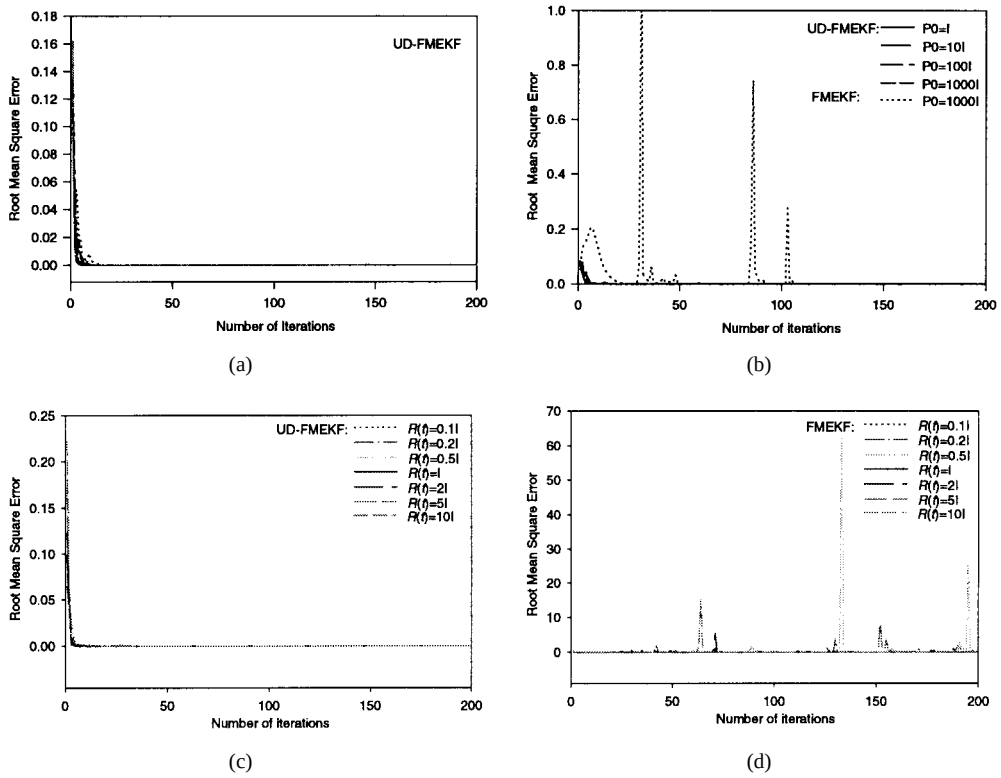

Fig. 4. Comparison of different start-up parameters for UD-FMEKF and FMEKF: (a) comparison of different initial weights, (b) comparison of different P_0, (c) comparison of different $R(t)$, and (d) comparison of different $R(t)$.

4) The Influence of Initial Weights and Covariance Matrix: Fig. 4 gives the RMSE curves of the UD-FMEKF and FMEKF algorithms with different initial weights, error covariance matrix $P(t)$ and noise covariance $R(t)$. It is obvious that UD-FMEKF was less sensitive to the changes in initial weights and covariance matrices than FMEKF. There were severe oscillations in Fig. 4(b) and (d) for FMEKF due to its poorer numerical stability. Note that the initial forgetting factor has to be chosen carefully to prevent the FMEKF from possible numerical divergence. The oscillations can be smoothed out more or less by reducing the required training accuracy so that the forgetting factor takes effect more frequently. The BP algorithm, however, is very sensitive to the initial choice of weights owing to its gradient-descent nature.

5) The Effectiveness of the Self-Adjusting Time-Varying Forgetting Factor: In order to verify the effectiveness of the self-adjusting time-varying forgetting factor, Fig. 5(a) and (b) shows the RMSE curves for different choices of the initial forgetting factors for UD-FMEKF and FMEKF, respectively. It can be observed that different factors mainly impact the convergence rate of the first ten time steps. However, the EKF obviously had a slower convergence than the FMEKF. Similar but better results were obtained by UD-EKF. Fig. 5(c) shows the RMSE curves for different specifications of learning accuracy using different thresholds for the adaptive adjustment of forgetting factors. In these simulations, the initial forgetting factor was $\lambda_0 = 0.99$, $\lambda(0) = 0.95$. It can be observed that the FMEKF was sensitive to the choice of the threshold, which implies that the threshold has to be carefully adjusted to obtain both fast convergence and high learning accuracy. As shown in Fig. 5(c), the RMSE's were quite different for different specifications of threshold, i.e., 10^{-2}, 10^{-3}, and 10^{-4}. A large oscillation appeared when the threshold was set to a value smaller than 10^{-4}. The learning accuracy would be poor if the threshold were chosen as 10^{-2}, although the curve would be smoother. In contrast, the UD-FMEKF was not sensitive to the choice of the threshold, and a much higher learning accuracy and much faster and smoother convergence could still be obtained even when the threshold was specified smaller than 10^{-5}.

6) Generalization Ability: As is well known, a large FNN with a single hidden layer is capable of approximating any continuous nonlinear function to within an arbitrarily small error margin over a compact region. This generalization property of the proposed algorithm for single hidden-layer FNN's was verified in this example. The predicted outputs using several sets of data samples that were not used for training demonstrate that UD-FMEKF has a good generalization capability. The demonstrated curves are omitted here due to space limitations.

7) Convergence and Computational Complexity: A comparison of the convergence rate and computational complexity for the UD-FMEKF, FMEKF, EKF, and BP algorithms required to reach the same learning accuracy is presented in Table II. It includes the required iteration numbers and computation time of the four algorithms for the accuracy thresholds of 10^{-2}, 10^{-3}, 10^{-4}, and 10^{-5}, respectively. The symbol "$-$" means the corresponding algorithm had

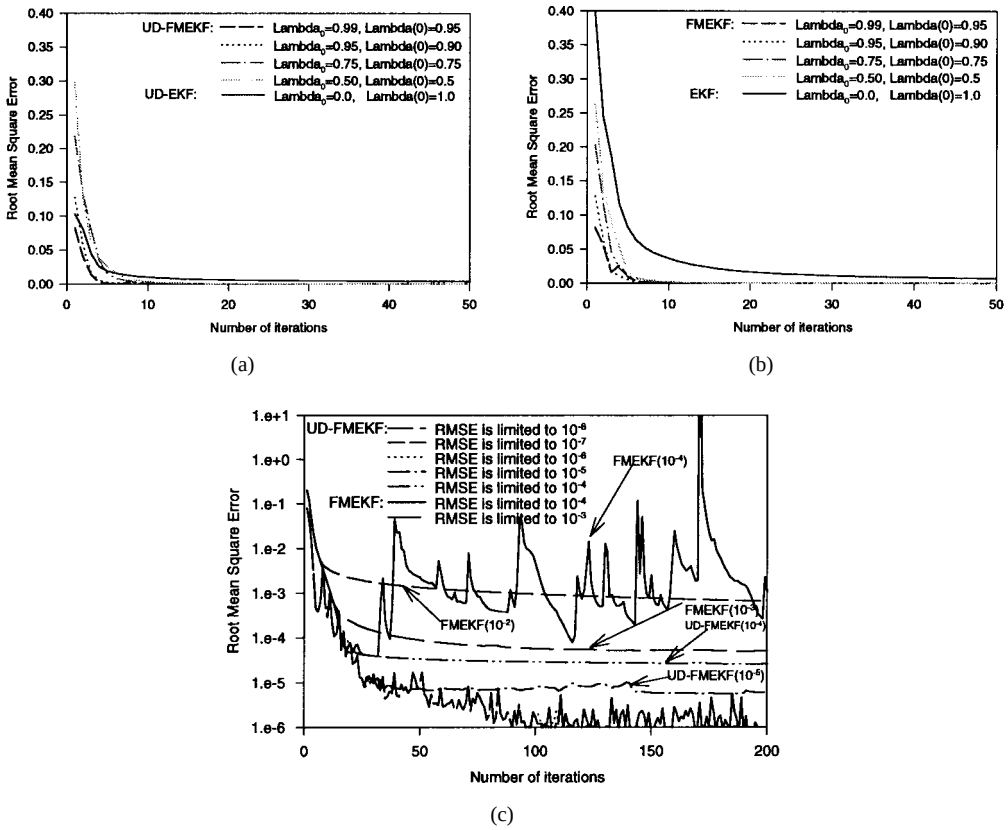

Fig. 5. Influence of accuracy thresholds and forgetting factors: (a) effect of forgetting factors on UD-FMEKF, (b) effect of forgetting factors on FMEKF, and (c) effect of accuracy thresholds.

TABLE II
COMPARISON OF THE ITERATION NUMBERS/COMPUTATION TIMES (SECOND) OF FOUR ALGORITHMS

RMSE	BP	EKF	FMEKF		UD-EKF	UD-FMEKF	
			Design 1	Design 2		Design 1	Design 2
10^{-2}	-	35/432	28/288	6/64	11/11	5/5	4/5
10^{-3}	-	-	162/470	12/127	-	18/18	5/6
10^{-4}	-	-	-	19/202	-	42/44	16/17
10^{-5}	-	-	-	-	-	79/82	26/26
10^{-6}	-	-	-	-	-	216/223	97/99

not reached the specified accuracy within 200 000 iterations for BP and 2000 iterations for FMEKF and UD-FMEKF, respectively. Table III lists the computation times per iteration for the three algorithms and the time-reduction ratio with UD-FMEKF and FMEKF for the case with different numbers of hidden nodes. It is obvious that a large time reduction was achieved as the number of hidden nodes increased for the UD-FMEKF.

It can be seen from Tables II and III that the UD-FMEKF gave the best convergence. Although the computation times *per iteration* for the UD-FMEKF and FMEKF were larger than for the BP, the computation time required by the UD-FMEKF for convergence to a specified accuracy is much smaller than for the FMEKF and BP. It further demonstrates the fact that UD-FMEKF is a highly efficient and effective algorithm for neural network learning in terms of computational cost and with respect to convergence rate and learning accuracy.

TABLE III
COMPUTATION TIME PER ITERATION VERSUS NUMBER OF HIDDEN NODES

n_h	BP(s)	FMEKF(s)	UD-FMEKF(s)	Time Reduction Ratio
10	0.035	10.62	1.03	10.3
7	0.03	3.93	0.52	7.6
5	0.02	1.54	0.27	5.7
3	0.015	0.42	0.11	3.8
2	0.01	0.16	0.05	3.2

8) Example 2: This example demonstrates the proposed algorithm's ability to identify the parameters of a nonlinear system [2]

$$y(t) = \frac{\theta_4}{1+\exp\{-(\theta_1 x(t-1)+\theta_2 y(t-1)+\theta_3)\}} + \varepsilon(t) \tag{23}$$

TABLE IV
IDENTIFICATION RESULTS OF THREE ALGORITHMS

Algorithms \ Parameters			$\theta_1 = 0.5$	$\theta_2 = 0.4$	$\theta_3 = 0.1$	$\theta_4 = 0.6$
$\sigma_x^2 = 1.0$	$\sigma_\varepsilon^2 = 0$	UD-FMEKF	0.5000000	0.3999995	0.0999994	0.6000000
		FMEKF	0.5000000	0.3999995	0.0999994	0.6000000
		BP	0.4999985	0.3999929	0.1000081	0.6000018
	$\sigma_\varepsilon^2 = 10^{-2}$	UD-FMEKF	0.5038201	0.3880315	0.1029965	0.5994509
		FMEKF	0.5038206	0.3880320	0.1029947	0.5986357
		BP	0.5080726	0.4369594	0.0738515	0.5986357
$\sigma_x^2 = 0.1$	$\sigma_\varepsilon^2 = 0$	UD-FMEKF	0.5000355	0.3999889	0.1000063	0.6000014
		FMEKF	0.5050219	0.4037437	0.0767170	0.5927105
		BP	0.5350003	0.2344401	0.0011955	0.5555543
	$\sigma_\varepsilon^2 = 10^{-4}$	UD-FMEKF	0.5002724	0.3986291	0.0978324	0.5991685
		FMEKF	0.4928689	0.3925616	0.1329743	0.6104050
		BP	0.5351063	0.228145	0.0010308	0.5549220

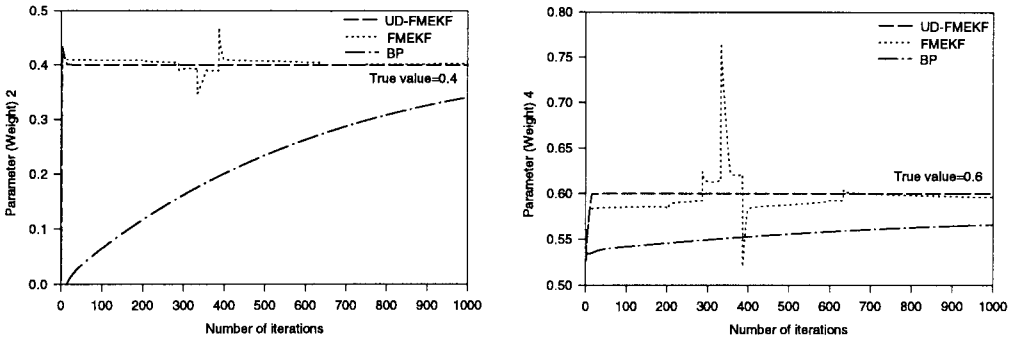

Fig. 6. Estimates of θ_2 and θ_4 for Example 2 (noise-free case).

where the input to the system $x(t)$ is an independent sequence of uniform random variables with zero mean and variance $\sigma_x^2 = 1.0$ (Case 1) or $\sigma_x^2 = 0.1$ (Case 2); $\varepsilon(t)$ is a Gaussian white noise sequence with zero mean and variance $\sigma_\varepsilon^2 = 10^{-2}$ (Case 1) or $\sigma_\varepsilon^2 = 10^{-4}$ (Case 2). The system parameters θ_i, $i = 1, \cdots, 4$ are equal to 0.5, 0.4, 0.1, and 0.6, respectively. The architecture of the FNN can be defined by two inputs, one hidden node and one output. Then we can define the input vector, $\mathbf{u}(t) = [x(t-1) \quad y(t-1)]^T$, and the parameter vector, $\boldsymbol{\theta}(t) = [\theta_1 \quad \theta_2 \quad \theta_3 \quad \theta_4]^T = [0.5 \quad 0.4 \quad 0.1 \quad 0.6]^T$ for the training algorithm.

The results of modeling and identification using the UD-FMEKF, FMEKF, and BP learning algorithms are given in Table IV and Fig. 6. It is obvious that the identification results provided by the proposed UD-FMEKF algorithm had the highest accuracy and converged to the true parameters. The FMEKF also had accurate final parameter estimates, except that there were oscillations because of FMEKF's poor numerical properties and sensitivity to the forgetting factor. The identification results of the BP algorithm were clearly not satisfactory.

The effect of the self-adjusting forgetting factor on parameter identification was also evaluated, but is not shown here due to space limitations. It is clear that for the FMEKF algorithm, an oscillation occurred when a learning accuracy of 10^{-5} or better is specified. The higher the accuracy specified, the faster the convergence, but oscillations may appear. For the UD-FMEKF algorithm, the satisfactory identification results were acquired even when the threshold was specified as 10^{-6}. For the BP algorithm, the results for different learning rates and momentum constants are obviously not satisfactory. It can also be observed that when the input excitation is large enough, accurate identification results for all algorithms are obtained. When the input excitation is smaller, however, the superiority of UD-FMEKF is more evident. In other words, when the system does not have the persistency of excitation or the persistent excitation condition is weak, the UD-FMEKF will provide much better identification results than the other two algorithms. We also simulated parameter identification in colored noise, and similar conclusions can be drawn as in the case with white noise. The best results were obtained by the UD-FMEKF. Detailed results were omitted due to limited space.

9) Example 3: Consider the following nonlinear MIMO system given in [6]:

$$\begin{bmatrix} y_1(t+1) \\ y_2(t+1) \end{bmatrix} = \begin{bmatrix} \dfrac{y_1(t)}{1+y_2^2(t)} \\ \dfrac{y_1(t)y_2(t)}{1+y_2^2(t)} \end{bmatrix} + \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix}. \tag{24}$$

TABLE V
COMPARISON OF THREE ALGORITHMS FOR EXAMPLE 3

Results		y_1		y_2	
Algorithms		Mean	s.d.	Mean	s.d.
Learning	UD-FMEKF	$3.5256039e^{-6}$	$7.3683759e^{-4}$	$2.3688755e^{-6}$	$1.2886386e^{-3}$
	FMEKF(SP)	$1.9358802e^{-4}$	$5.8601208e^{-3}$	$-9.4661071e^{-5}$	$6.6658701e^{-3}$
	FMEKF(DP)	$2.8967550e^{-6}$	$1.1308955e^{-3}$	$3.7317728e^{-6}$	$1.6937216e^{-3}$
	BP	$2.0058365e^{-2}$	$1.3416174e^{-1}$	$1.3929651e^{-3}$	$6.7967173e^{-1}$
Generalization	UD-FMEKF	$-3.7519666e^{-3}$	$1.8191844e^{-1}$	$-1.0500047e^{-2}$	$1.8922930e^{-1}$
	FMEKF(SP)	$-3.1498377e^{-2}$	$2.1606126e^{-1}$	$-6.1361555e^{-2}$	$2.3214535e^{-1}$
	FMEKF(DP)	$-4.3691970e^{-3}$	$1.7857307e^{-1}$	$-2.9752850e^{-2}$	$1.9580073e^{-1}$
	BP	$-1.9512380e^{-1}$	$8.7109413e^{-1}$	$1.2051304e^{-1}$	$1.0580563e^{0}$

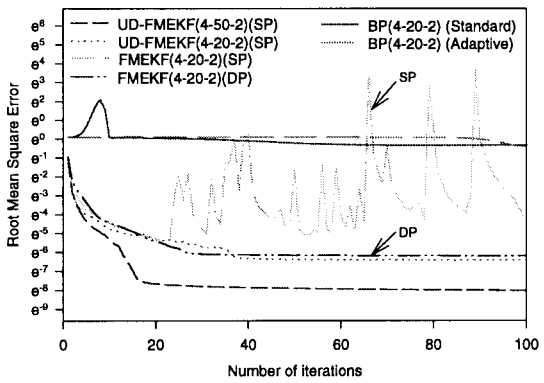

Fig. 7. RMSE comparison of different algorithms.

An FNN of a 4–20–2 structure was used to model and identify the above MIMO system. The RMSE curves of the three algorithms for a vector input $[\sin(2\pi t/25), \cos(2\pi t/25)]^T$ over 100 learning iterations given 500 training data samples are shown in Fig. 7. Table V shows the mean and standard deviation (s.d.) of the learning and generalization output errors of the algorithms. The results obtained for the BP were computed using the fast adaptive learning rate BP algorithm, given in the MATLAB Neural-Network Toolbox 2.0, which is much faster than the standard BP. It is clear that the proposed UD-FMEKF had the fastest convergence and the best accuracy. FMEKF had faster convergence than BP. Due to the increase in state dimensions and poor numerical stability, FMEKF often exhibited severe oscillations and sometimes even diverged. As such, a double precision version had to be implemented to prevent the possible numerical divergence. From Fig. 7, it can be seen that the double precision (DP) version of FMEKF had some RMSE's slightly larger than the UD-FMEKF with single precision (SP). The learning results of "standard" and "adaptive" learning rate BP algorithms were also calculated for comparison with the UD-FMEKF and FMEKF for the same zero initial condition. Very slow convergence and poor generalization results were obtained by the BP algorithms. If the initial weights were selected randomly, slightly better results were obtained. This fact also indicates that the BP algorithm is sensitive to initial weights, whereas the UD-FMEKF and FMEKF are not. This example further demonstrates the superiority of the UD-FMEKF algorithm proposed in this paper.

V. CONCLUSIONS

Two fast learning algorithms for training FNN's by using a FMEKF and a U-D factorization-based FMEKF, combined with a self-adjusting time-varying forgetting factor, have been proposed in this paper. It has been demonstrated that the proposed UD-FMEKF algorithm with the adaptive forgetting factor greatly improves the convergence rate, numerical stability, and accuracy with fewer hidden nodes. In addition, this new algorithm is less sensitive to the choice of initial weights and covariance matrix, and the variation in the observed data. It has also been demonstrated that the proposed algorithm can be used successfully for modeling and identification of highly nonlinear systems. The algorithm proposed here is not restricted to networks of a specific topology. This algorithm or a variation of it has been used effectively for system modeling and identification through the use of radial basis function networks [12] and functional-like neural nets [13]. This algorithm can also be used for training other variants of FNN's and recurrent networks and with other training algorithms [11]. The proposed algorithm can also be readily extended to the decoupled (or multiple) EKF formulation, with each of the decoupled filters using the UD-FMEKF, to reduce the computation requirement in the training of large-scale systems.

ACKNOWLEDGMENT

The authors would like to thank Prof. K. S. Narendra for his helpful discussion and suggestions. Thanks are also due to the reviewers whose comments significantly improved the presentation of the paper.

REFERENCES

[1] G. J. Bierman, *Factorization Methods for Discrete Sequential Estimation.* New York: Academic, 1977.
[2] S. A. Billings, H. B. Jamaluddin, and S. Chen, "Properties of neural networks with applications to modeling nonlinear dynamical systems," *Int. J. Contr.*, vol. 55, no. 1, pp. 193–224, 1992.
[3] G. Chen and H. Ogmen, "Modified extended Kalman filtering for supervised learning," *Int. J. Syst. Sci.*, vol. 24, no. 6, pp. 1207–1214, 1993.
[4] Y. Iiguni, H. Sakai, and H. Tokumaru, "A real-time learning algorithm for a multilayered neural network based on the extended Kalman filter," *IEEE Trans. Signal Processing*, vol. 40, no. 4, pp. 959–966, 1992.
[5] L. Ljung and T. Soderstrom, *Theory and Practice of Recursive Identification.* Cambridge, MA: MIT Press, 1983.

[6] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Networks*, vol. 1, pp. 4–27, 1990.
[7] D. Obradovic, "On-line training of recurrent neural networks with continuous topology adaptation," *IEEE Trans. Neural Networks*, vol. 7, pp. 222–228, 1996.
[8] G. Puskorius and L. A. Feldkamp, "Neurocontrol of nonlinear dynamical systems with Kalman filter trained recurrent networks," *IEEE Trans. Neural Networks*, vol. 5, pp. 279–297, 1994.
[9] S. Shah, F. Palmieri, and M. Datum, "Optimal filtering algorithms for fast learning in feedforward neural networks," *Neural Networks*, vol. 5, pp. 779–787, 1992.
[10] S. Singhal and L. Wu, "Training feedforward networks with the extended Kalman algorithm," in *Proc. Int. Conf. ASSP*, 1989, pp. 1187–1190.
[11] Y. M. Zhang and X. R. Li, "A fast U-D factorization-based recursive prediction error learning algorithm for feedforward neural networks," in *Proc. 35th IEEE Conf. Decision Contr.*, Kobe, Japan, Dec. 1996, pp. 2036–2041.
[12] ———, "A hybrid training algorithm for RBF networks with application to modeling and identification of nonlinear systems," in *Proc. 35th IEEE Conf. Decision Contr.*, Kobe, Japan, Dec. 1996, pp. 937–942.
[13] ———, "Helicopter parameter identification using functional-link neural net," in *Proc. 2nd Chinese World Congr. Intell. Contr. Intell. Automat.*, Xian, PR China, June 1997.